

May 1988

Vol. 1 N° 8

Price £1.50

Archive

The Subscription Magazine for Archimedes Users

Using Anti-aliased Fonts

Sound Synthesis Program

EMR SoundSynth Review

Playing Music in BASIC

FORTTRAN Graphics Library

Two new regular columns: MS-DOS and Languages

Free Pull-out Section: Index to Archive issues 1 – 6

Games and Art Packages Reviewed

Amateur or Professional?

Thanks very much indeed to all those who have contributed material for Archive magazine. As I have said before, the standard of the magazine depends very much on these contributors. But as Archive expands (we now have over 1,500 subscribers) we have had to think whether it is time to start paying contributors for their articles, as other professional magazines do.

Our feeling is that the whole atmosphere of this magazine is based on the free flow of information – everyone trying to help others by giving out as much information as possible instead of trying to withhold it in order to make money.

In one sense, Archive is a 'professional' undertaking as we do need to earn enough to live on and we try to be 'professional' in terms of our standards of production and the service we provide. Never-the-less, we still want to retain the 'User Group feel' and, to that end, we have turned down all the offers we have had which would have involved marketing software or hardware under the Archive name.

So, what we will do is make as much information as possible freely available to you, the reader. What we ask you to do, in return, is to continue to send in your hints, tips and longer articles and continue to buy your commercial software and bits of hardware through us to help finance the project as a whole.

Many thanks again and here's wishing you every success in your own use of the Archimedes computer.



P.S. Why not come and see us at the Micro User Show, May 13th – 15th, New Horticultural Hall, Westminster?


Archive

Volume 1 • Nº 8 • May 1988


Contents

Hardware & Software Available ...2	FORTRAN Graphics Library20
Matters Arising2	Bug or Feature?.....21
Comment Column3	MS-DOS Column22
Readers' Comments5	Index of Issues 1 – 625
Help!!!.....6	Using the Debugger – Part 2.....29
Hints & Tips7	Beginner's Guide to Fonts37
Contact Box8	Art Packages Review44
Information about Monitors8	Games Review47
Sound Synthesis Program9	Fitting the Hard Disc Upgrade48
Playing Tunes in BASIC.....13	Base Conversion Program50
Errata.....15	Small Ads52
EMR Sound Synth Review16	Fact-File53
Languages Corner18	

Hardware & Software Available

- **SigmaSheet from Minerva** is claimed to be 200 times faster than Intersheet, Lotus 123 and Logistix and capable of handling sheets over 7 times the size of Lotus 123. It should be available off-the-shelf by the time this magazine gets to you. The full price £69.95, Archive price £64. We hope to be able to review it next month.
- **Two new games**, also from Minerva are: **Hoverbod** and **Missile Control**, £14.95 each. They should be available "in May". More details next month.
- **A screendump for 24-pin printers** (LQ800 and LQ1000) available from Abacus Training. The dump utilises the triple density graphics mode to produce a picture of 1280 x 1024 dots which comes out at about 7.5" x 5". This is was written by Mark Sealey and Gerald Fitton, two of our contributors, and is being made available to Archive subscribers for £5. (Or £3.50 if you send a formatted disc.)
- **U-Connect** from Magenta Research – a general purpose comms package, exclusively WIMP driven, providing various terminal emulations plus a selection of file transfer protocols: X-modem, Kermit and CET for Prestel. Will run on 305's, but "1Mbyte preferred". £59.95 inclusive. We've been promised a review copy, so we hope to report on this in a future issue.
- **ArcTerm** – a free downloadable program from Hugo Fiennes on his bulletin board on 0458 476081. (How do you get it if you haven't got a comms program for your Archimedes?)
- **PipeDream** (alias View Professional) is being launched for the Archimedes by Colton Software at £99 +VAT "from May 19th". Pipedream is an integrated suite of software: word-processing, spreadsheet + database. More details from Robert Macmillan on 0954 211472.
- **Database of Educational Software and Addresses.** This database, already well established on Viewstore, FIND, Supastore and KEY for the BBC B and Master series is now available (well, it said it should be by mid May) on System Delta Plus for the Archimedes. £20 from Nick Evans, P.O.Box 55, Grimsby, DN32 0QB. Educational orders welcome.
- **Fast action arcade game: QUAZER** very colourful, very fast, several levels of skill, but basically just shooting up the on-coming enemy hordes. (My kids like it!) £8.95 from J. Rockey, Brecklands, Broad Oak, Shrewsbury, SY4 3AH or send S.A.E for information. 

Matters Arising...

- **Bush Rescue** from 4mation. I didn't read the press release that Acorn sent out carefully enough... Bush Rescue is for the BBC micro and Master series computers, but not the Archimedes. Sorry about that!
- **File paths:** On page 10 last month; The "previous directory" is "\", not "}"!
- **Local DATA statements:** We are assured that version in BASIC V version 1.03 (not yet released), RESTORE can restore relative to a line, so DATA can be put in any sort of program, including libraries. Also, the DATA pointer can be made LOCAL and RESTORED like the error handler. The only problem is when will it be available, and how much will it cost? In the meantime, for an improved version of local DATA statements, see Joc Carlton's music program on page 13.
- **System Delta Plus.** In Hints & Tips last month, we suggested that if you had problems with "No Room", you should add ".D" to the end of the filename – it should have said ".D". 

Comment Column

Eureka! Another first for Archive

Archive was the first professionally produced magazine for the Archimedes and now we have launched the very first Bulletin Board specifically for the Archimedes.

At least, I hope that by the time you receive this magazine, Archive will be on-line as the "Eureka! The Bulletin Board". If you have got your RS423 working and have a suitable modem (1200/75 or 300/300) ring in on 0603-250689 and access our new Bulletin Board. Facilities are at 3 levels:

- **General Access** – anyone can call in and get a sampler of Archive magazine – articles to read and one or two programs to down-load.

- **Archive Subscribers.** If you are armed with your subscription number as a password, you can get further facilities:

Next month's news – snippets of what is to come in the next issue of Archive

Message board – leave messages for other subscribers perhaps asking for or offering help.

The Archive editor and the editors of Eureka, Carl Wright and Paul Wickham, plus some of the other contributors will have mailboxes, so you will be able to leave messages for them.

- **Eureka! Subscribers (£5 / qtr).** They will have their own personal mailboxes and will also be able to download the program listings from all the issues of Archive magazine as well as other programs which subscribers have sent in.

If you have programs (preferably ones that are self-documenting) that you would be prepared to put up on Eureka!, let us know. Please test them very thoroughly before you send them because we want to maintain our reputation for quality.

Yes, you guessed it, although SID really is the name of the Acorn Bulletin Board, "CHARLIE" was a spoof – well it was in the April issue! Mind you, one

or two ingenious folk managed to suggest a solution to the acronym. My favourite is Combined Help And Response Line for Inquisitive Enthusiasts!

"Can I help?"

Since the plea in last month's issue for help in editing articles, we've had a lot of offers of help. The largest number were offering to help edit the languages section, then MS-DOS with only couple of people interested in helping with Econet. Within each group, I will try to put them in touch with one another so that they can then share ideas and between them, they will hopefully come up with a series of articles. I have suggested that they take it in turns – perhaps two or three months each – to be the actual editor to co-ordinate the material and feed it in to Archive.

Brian Cowan has started us off on the Languages front and Ken Biddle on MS-DOS applications.

Calling all enthusiasts!

If you have a special interest in a particular subject and want to be put in touch with others with similar interests, just send us a contact and we will publish it so that others can get in touch with you. (You could also use Eureka!) See page 8 for more details and the first contact to be sent in.

ArcWriter still free?

The rumour we hear is that Acorn have changed their minds and are going to continue to give a copy of ArcWriter free with each Archimedes. (Can anyone who has recently bought either a 300 or a 400 series machine, confirm this?)

RAM prices climbing

You may well have heard that Acorn, along with other computer manufacturers, are having to consider increasing the prices of their computers because of the world-wide shortage of RAM chips and the consequent rise in prices. There is no definite news of a price rise, but it may be the spur that some of you need to take the plunge and buy your Archimedes now. You may perhaps want to use it as extra ammunition in trying to persuade those who hold the purse-strings (bosses or wives or

whoever!) that it would be prudent to buy now before the prices go up.

Podule RAM prices rise

The general rise in the price of RAM chips has also affected those who have already bought their Archimedes – the RAM chips for the ROM/RAM podules have also gone up in price. Computer Concepts have had to put their price up to £10 + VAT. We managed to get 50 chips from another supplier and were selling them at £11 (inc VAT). However, these have all gone now and we are buying them direct from Computer Concepts again. They can only give us a limited discount, so we will have to sell them at the same price – £11.50 including VAT.

PC Emulator

Acorn are saying that “The new emulator offers better speed, better compatibility and more memory free to PC applications, almost totally regardless of the configuration of your machine. In addition, a new manual is supplied which has the benefit of documenting PUTFILE and GETFILE”.

One cannot help but smile at the last comment. Acorn are recommending the new version because now, the manual actually explains certain of the commands which the software provides!

I understand that the program discs that they are sending out now, contain versions 1.00, 1.09, 1.20 and 1.21! The 1.20 version is described by Acorn as “the current version” and the 1.21 as “under development”!

What price the Archive discs?

“What are the Archive program discs supposed to be?” and “Why don’t we put more on them?” – these are two questions I am asked occasionally.

Well, it was never intended to be more than a machine readable copy of the programs in the magazine and, as such, the price of £3 is quite reasonable, I think. We do not have the time or programming resources to put auto-menus and various other bits and pieces on it. However, if we have odds and ends of things which may be useful, we’ll put them on the disc provided you accept them

for what they are – i.e. if they are useful to you, great, but if not, please don’t feel that you are getting sub-standard goods – what you paid for was the programs from the magazine – the rest is the equivalent to the plastic submarines you get with the Cornflakes!

Having said all that, there are one or two extras on this month’s disc. Because of the emphasis we seem to have placed on sound this month (I say seemed because I can only publish the articles I get sent each month) I have put in some User Tunes for playing with the Music Editor. Not all of them are complete, but just take them as you find them. If you have done some fantastic tunes and want to spread them around, by all means send them in and we will put them on next month’s disc.

Wot?! No adverts?

If you look through this month’s magazine, you will find that there are no adverts. Does this represent a change in Archive’s policy? No, the main thing it reflects is that no-one has sent any adverts in! We don’t have time to chase potential advertisers and so inertia wins and they don’t bother advertising.

Mind you, with the continuing rise in the amount of material we are being sent, it’s a good job we didn’t have any adverts – we needed all the space for articles! In fact, on some articles, to fit it all in, I’ve reduced the point size of the text slightly. The text in most of the articles is one point size smaller than we’ve used before. (The article on page 37 is in the size we used previously.) If you think it’s getting a bit too small, let us know.

No Acorn software reviews?

Many of you have noted that we have not, as yet, done any reviews of Acorn software (apart from Fortran 77). The reason is simple. Despite my having asked a number of different people, Acorn have not sent a single piece of review software to Archive magazine! I am tempted to conclude that our policy, from now on, should be only to review software from those suppliers who are prepared to support us by sending review material.

If you have any views about this or any of the other matters raised in Archive, please let us know. **A**

Readers' Comments

• **More about DATA statements for BASIC** programs without line numbers... Gerald Fitton writes... now that we have easy access to disc files, there is no justification for keeping data within a program. It is much better technique to keep the data in a separate file. If the data does not change very often then it could be declared as a global variable (e.g. month\$= "January, February, March,...") at the start of the program or as a LOCAL variable for use within the LIBRARY function or procedure. If it might need to be changed (e.g. the notes of a tune) then it is better *LOADed as a data file into a byte array (which is my preference for small data files) or selections from it can be INPUT# (or BGET#) into a string, integer or real array or variable.

• **PC-Emulator.** We have had several irate letters from readers about the PC emulator. The gist of the comments is that they think it is awful that after paying £115 for the emulator, which has no manual for either MS-DOS or GW-BASIC, (as you get when buying the same software for an IBM machine) they then have to pay the extra £15 for the up-grade to 1.09.

One person writes... Now, there is a 1.20 PC emulator, superseding the 1.09 version for which I have already paid £15 extra – it is apparently faster and more compatible than 1.09, **BUT** they want **ANOTHER £15** off me to do the up-grade. Apparently, the £15 is an 'administration fee' for copying your disc and posting it back to you. OK, so in a big company it costs money to put cheques through the accounting system but surely the answer is **not to charge at all** but insist that people send a return envelope or sticky label and postage. It can't take more than 2 minutes of someone's time to copy a disc and shove it back in an envelope, can it? Is the £15 a time that Acorn get out of it **REALLY** worth the aggravation it causes? I very much doubt it. Another case of Acorn shooting itself in the foot!

• **Watford 5.25" Disc drive interface** – One or two folk seemed not too pleased with Watford's interface, but I understand that Watford are trying improve on their first efforts.

• There are reviews of ArcWriter and Clares' GraphicWriter in several places, most recently in Acorn User for May, and Mike Bailey wants to set the record straight.

I think there is unnecessary bias against ArcWriter. Yes, it has too many bugs and not all of them are in the relatively unimportant area of failing to rewrite the display – one can always scroll away and back. Most important for me is that it is too slow. It can't keep up with my 'power typing' and when the keyboard buffer gets full it jst lses lettrs. (sic) (Actually I'm using TWIN now, as I do with large volumes, and then I import the text into ArcWriter).

With ArcWriter, I get very nice NLQ out of the Panasonic KXP1081, using bold, underline, italic, and the rest. I had to use the offline printer code program, but it let me do all the fonts of ArcWriter in one file. The supplied definitions for FX80 and LX80 got me going well. Now that I'm used to ArcWriter, I don't seem to hit the bugs.

Recently I bought GraphicWriter, partly on the strength of the reviews. I was immediately disappointed that it is not a true WYSIWYG processor, whereas ArcWriter is. But I gave it a fair crack, and tried migrating a 20 page script to it. So I exported it from ArcWriter and imported it into GraphicWriter. Not trivial – the GraphicWriter manual is far worse than it's given credit for in the reviews ("the 63 page manual is abysmal" says Acorn User).

The files panel is very confusing and it took me ages to get the various pieces into the multi-file structure. I lost everything once and had to start again. When I started working through specifying the fonts, I found that it only supports half the fonts that ArcWriter does. There is a note in the book saying you can use the offline printer definition program to make bold underlined come out as italic, elite or some such, but the maximum number of fonts is still only half. Acorn User says of GraphicWriter, "The wordprocessor allows you to use all the normal range of effects..." and later of ArcWriter, "The range of effects is adequate, though hardly exciting." That's plainly unfair.

Now to a nasty design feature of GraphicWriter (I'm learning not to call these things bugs. It seems that the graphical page make-up is done by physical page number, of which there is only one set even for multi-file documents. After carefully putting a box around a heading on page 2 of document 1, I found the same box appeared in the same place in document 2. Quite a shock, as I'd been hoping to use the multiple document print feature later – now I would have to sit there making up the pages for one document at a time and printing it. GraphicWriter doesn't save the graphical make-up either.

Another fault is a subtle one about how lines are scanned and converted to bit-images for the printer. I can have lines on the screen which all look the same width but come out on the printer with a variety of widths.

In summary, I think Clares have missed the point. If you go for What You See Is What You Want (WYSIWYW) then you should choose a scheme like IBM's GML, which I have used for years, where the controls are human readable and high level, unlike GraphicWriter which is neither. The GML way, the user can add function by writing more SCRIPT macros and get some very sophisticated DTP functions. I expect ArcWriter is extendible by adding fonts and Acorn may eventually do it. GraphicWriter seems to be a dead-end design. I may still use it a bit, but it has less usability and less function than ArcWriter. I agree with Acorn User that neither product is good, but I look forward to an ArcWriter that performs properly, with the bugs fixed. Then I would happily part with £29.95 for it. **A**

Help!!....

• **Arc-Writer:** (T.P.Whitehorn writes) You can transfer documents between discs by clicking on the 'A' icon and *MOUNT'ing the new disc and then INSERTing the document into a new document file. This works once, but then the program crashes – at least it does on my 310. It seems a primitive method. Is there any other way of doing it?

• He continues... Is there any chance of some articles on **3-D graphics for beginners** – or is that a contradiction in terms? Any offers? (*Have a look in RISC User, issue 5! Ed.*)

• **Could we have some basic (not BASIC) articles**, says one reader, for people who are mind-boggled by 'configure', 'screensize', 'modules', 'SWI's' etc. Especially perhaps with those people in mind who are NOT ex-BBC users. Any offers?

• Geoff Clout from Leeds asks whether anyone knows of a good **label-printing program**, in particular for printing audio cassette labels on an Epson compatible. (Native mode or BBC under 65Arthur would do.)

• **CGP220 screendump.** David Palmar of Jordanhill School asks whether anyone has a screen dump for a Tandy CGP220 colour printer.

Help answers...

• There is a **partial renumber** built into BASIC which is used to renumber APPENDED programs, however in general it's just not worth the hassle of trying to get at it! In days of yore there were 6502 programs that did partial renumpers published in magazines: the data format of a program is identical, so perhaps they could be used under the emulator? (Someone has just sent in a BASIC PROCEDURE for a partial renumber. I'll try it out first before publishing it.)

• **Passing variables between a suite of programs.** One suggestion is to push PAGE up or HIMEM down and use indirection operators, ?, !, | and \$ for byte, four-byte, five-byte and string variables respectively.

• **File transfer with Apricot.** Tom Fortescue has provided the enquirer with a program listing. I suspect that if you sent Tom an A5 S.A.E. he could be persuaded to send the same comprehensive documentation to other folk but I haven't had time to ask him, so either ring us or send your S.A.E. via Archive magazine. **A**

Hints and Tips...

• **Colour TV output for Archimedes?** If you haven't got a colour monitor and want to use a TV to look at the output from the Archimedes in colour, you can feed the signal from the video output into the 'video in' on a video recorder (always assuming you've got one!) and connect the 'RF out' from the recorder to the TV.

• **Drive lights on external 5.25" drives.** One reader suggested that to avoid the external drive light coming on when I accessed the internal drive(s) I should remove all links except DS1 (I use DS2 to make it drive 2), TO, RR and MS. I followed his advice and it seems to be OK now.

• **400 or 300?** If a piece of software wants to find out if it is running in a 300 series machine or a 400 series it can read the MEMC to find the size of page being used, thus:

```
SYS "OS_UpdateMEMC",0,0 TO size
IF (size AND %1100) = 0 THEN
    PRINT "I'm a 300!"
ELSE
    PRINT "I'm a 400!"
ENDIF
```

• **More buzzing.** One reader noticed that the buzzing from the speaker gets worse when you upgrade from a 305 to a 310. Other readers have found that the buzz gets to an intolerable level with certain software such as ArcWriter. In any case, this is now recognised by Acorn as a field change, so your local dealer should be able to sort it out for you. If your local dealer is not within easy reach then the "capacitor fix" mentioned in issue 3, page 7, is easy enough to do as long as you are reasonably competent at soldering.

• **Printer Acknowledge Line.** The IOC (Input Output Controller) has a printer acknowledge line which can be read from the ARM supervisor mode. However, it is not advisable to 'play around' in this area, unless you know what you are doing, because some locations are read only and attempting to write to them could damage your IOC chip. Here is an example program that reads the printer acknowledge line.

```
10 REM >$.PrintAck
20 DIM code% &100
30 DIM flag% 1
40
50 FOR opt=0 TO 2 STEP 2
60     P%=code%
70     [OPT opt
80     .printer_ack
90     SWI "OS_EnterOS"
100    TEQP PC,#3
110    MOV R0,#&3200000
120    ADD R0,R0,#&10
130    LDRB R1,[R0]
140    AND R1,R1,#1
150    ADR R0,flag%
160    STRB R1,[R0]
170    TEQP PC,#0
180    ORRNE R0,R0,R0
190    MOV PC,R14
200 ]
210 NEXT opt
220
230 MODE 0:OFF
240 PRINTTAB(10,5);"Printer is ";
250 REPEAT
260     CALLprinter_ack
270     PRINTTAB(21,5);
280     CASE ?flag% OF
290         WHEN 0 :PRINT"not ready."
300         WHEN 1 :PRINT"ready.    "
310     ENDCASE
320 UNTIL INKEY(0)=32
330 END
```

• ***TypeFile Command.** One reader tried to define an alias which types out a file on the printer and then switches the printer off again, but he found that it was easier said than done. I set Adrian Look onto it and has managed to find a way of doing it (albeit rather tortuous!):

```
*SET Alias$TypeFile ECHO
<60>2<62>|M TYPE %0|M ECHO
<60>3<62>|M
```

then to print out the file, you do a *TypeFile

<filename>. The <60> and <62> are the ASCII codes for "<" and ">".

It sounds a bit long-winded, but what you are trying to do is generate ASCII codes <2> and <3> to switch the printer on and off again. When you do a *SHOW, it gives the definition of TypeFile as ECHO <2>IM TYPE %0IM ECHO <3>IM.

• "What's the time, Arthur?" – If you wanted to change the prompt which Arthur gives, you could try, for example:

```
*SET CLISPROMPT >>>
```

and you would get a question mark instead of the star prompt. Then, if you want Arthur to tell you the time at each prompt, use:

```
*SETMACRO CLISPROMPT <SYS$TIME>*
```

This prints out the time and then prints the star prompt. The reason you use SETMACRO rather than just SET is that it needs to be a variable which is up-dated each time the command is used. If you don't like the seconds figure to be included, try:

```
<SYS$TIME><127><127><127>*
```

This prints out the time but then generates three delete characters which remove the seconds figures and the colon. **A**

Contact Box

This is a new section in Archive. The idea is that if you have a particular interest in using the Archimedes and want to contact other people interested in the same sort of thing, you can send in your name and any or all of the following: address, phone number (work or evening) or mailbox number. Other people can then contact you to express their interest and you can form some sort of closed user group.

As I understand the Data Protection Act, I can't give out addresses of subscribers to other subscribers, but if you are prepared to be a contact point, people can get in touch with you. David Palmar starts us off this month...

• **Geography Teachers** please contact David Palmar, Jordanhill College School, 45 Chamberlain Road, Glasgow, G13 1SP. **A**

Monitors' List

• Roger Wilson from Acorn writes... I've used both **NEC** and **Taxan** monitors for some time. The Taxan has many advantages since it can be adjusted to match Arthur (See below) but the NEC picture is definitely better.

• **Centring the Taxan display.** Rabin Ezra writes... It is possible to get the display on the hi-res modes centred on the screen. (See complaint in the comparative review last month.)

1. Set all push button switches to out. (i.e. select colour, analogue, auto, auto, auto.)
2. Set knobs for V.size, V.posi, H.posi to their centre click positions.
3. With a low resolution mode selected, adjust the presets in row C to give an acceptable picture.
4. With a hi-res mode selected, adjust the presets in row B to give an acceptable picture.

• **Phillips CM8873** – Adjustments to correct the sideways shift are available on the p.c.b. inside it. Service manuals are available from Phillips.

• **Fujitsu ME-503 multi-sync monitor** is on sale from Viglen at £343.85 (inc VAT) + £8 carriage – substantially less than the £544 we were quoting for the NEC Multi-sync II. Does anyone know if it is any good? I asked Viglen if we could have a look at one but they said they were not interested in letting us borrow one for review "at this stage".

• **Kaga Vision III.** We are told that, yes, it does work, but as far as I can gather, only on TTL, i.e. it will not give the full range of colours unless you get or make a suitable adaptor.

• **Mitsubishi EUM 1471a.** One West German reader tells us that it is comparable with Taxan 770 Plus and NEC MS II but that it has an 18mm shift in modes 18-20 and a bit of distortion in other modes.

• **WARNING.** One reader tells us **NOT** to use *Configure MonitorType 2 unless you have a monitor suitable for modes 22 and 23. He says this because, by doing so, he damaged his monitor. (He didn't say what the damage was or how much it cost to repair, but "better safe than sorry".) **A**

Sound Synthesis – A Detective Story!

Ian Nicholls

By now, you will no doubt have played a bit with the Music Editor on the Welcome disk, listened to "TUNE1" and "TUNE2" and maybe even entered a tune or two of your own. However, the magic may have worn off a little after you realised that there was no easy way to add to the selection of string and percussion voices included in the system ROM! Indeed, there is no mention of this subject whatsoever in the, otherwise, impressive documentation that Acorn send you with the Archimedes.

If you have bought the Archimedes "Programmers Reference Manual" (PRM), the chapter on Sound will not have been the easiest chapter that you have read, but it contains hints of features that might provide a clue to expansion of the range of voices. It tells you that the library of internal voice generators (i.e. those in the system ROM) includes simple waveform oscillators and a speech data interpolator, plus the string and percussion voices.

Now before you get too excited about this speech business, Acorn point out in the Addendum to the PRM that the speech data interpolator is not, after all, in the library of internal generators! That still leaves the "simple waveform oscillators". The only sound generator left in the system ROM is "WaveSynth". The waveform oscillators and WaveSynth must be one and the same!

WaveSynth

Delving further into the PRM Addendum reveals that you can have multiple copies of a relocatable module with version 1.2 of the Arthur operating system. Such copies, or "instantiations" as Acorn call them, have many workspace areas but only one code area. The example quoted is, yes you have guessed it, WaveSynth, with WaveSynth%Brass being the name of the instantiation of the WaveSynth module with the Brass voice loaded.

Believing myself to be on the trail of something interesting here, I disassembled the WaveSynth module looking for evidence of how to load a new

voice. I came across an undocumented SWI call, which appeared to do just that. However, by this time the trail of clues was running out (as was my time!) so, more or less in desperation, I wrote to Acorn. I received a letter in reply, enclosing a disc containing a new voice for me to try (in the form of a wavetable), the program that generated it, and the data for the BEEP wavetable. (BEEP is the name of the sound that WaveSynth normally produces – the "bong" Archimedes makes when you switch it on.)

At the end of the article, parts of this program are reproduced, enough in fact for you to create the new voice for yourself. In the next article a much more generalised version of the program will be given which will enable you to create an almost infinite variety of new voices. I will also give you the missing parts of Acorn's original program and pose one or two questions to which I do not have the answer. You can be the ace detective instead of me!

What I will be describing in this and the succeeding article are software features of the Archimedes sound system which Acorn are not prepared to support at present. I will return to Acorn's position on all of this later, but first I need to explain what a wavetable is and how you use one to create sounds.

Wavetables

All electronically-produced sounds reach our ears via a loudspeaker. The loudspeaker makes a sound as a result of a varying voltage being applied to it. In almost all of the micros prior to the Archimedes (including the BBC micro), this varying voltage has been generated directly by a special purpose sound chip. Such chips typically provide only one shape for the varying voltage, and hence for the resulting sound, although software allows the user to alter the frequency of the sound, its volume and the length of time for which it sounds. The most usual shape for the varying voltage (because it's very easy to generate with digital circuitry) is a square wave, which alternates between two values. However, sine waves are the basic building blocks of almost all sounds and the program below uses them to create a sound something like a trumpet.

The approach adopted in the Archimedes is entirely different. It has no sound-generating chip but instead it calculates the shape of the sound-wave entirely in software! A digital to analog converter (DAC) then converts the resulting stream of numbers into the corresponding varying voltage. This whole process is only possible because of the immense speed of the ARM RISC processor chip.

All musical sounds have a repetitive waveform: if a piano plays middle C, which has a frequency of approximately 262 Hz, the sound we hear is made up of the same waveform being repeated 262 times a second. If we know the shape of this repeating waveform, we can calculate its value at many equally-spaced points during one cycle and store them in a table. To recreate the waveform and generate a sound, we just need to write a routine which repeatedly reads the successive values in the table and sends them to the DAC. When the end of the table is reached the routine begins reading from the beginning of the table again.

If we read values from the table at the same rate as before, but read only every second value, then we will move through the table twice as fast and the resulting sound will be at a higher pitch, in fact twice the frequency. Similarly, if we read every value twice, it will take twice as long to step through the table and the resulting sound will be half the original frequency. So, by storing only one table (a wavetable), and altering the way we read values from it, we can create a whole series of similar-sounding but differently-pitched notes. If you want to explore these ideas a bit further then the references at the end of the article are worth reading: I would particularly recommend the two articles by Hal Chamberlin.

Creating A New Voice

The program below is based on this method of generating a sound. In fact, it goes much further than creating just one wavetable, it creates and stores 330 of them, and each one contains 256 separate values. The reason for having so many of these wavetables is to take account of the variation of the loudness of the sound (a very rapid build-up, with a more or less constant loudness in the middle and then a decay to nothing), and the changing mix

of harmonics as well (we will look at these in the next article). The program that reads the 330 wavetables is WaveSynth and every one hundredth of a second it moves on to the next one, so the note that is produced will last 3.3 seconds if it is allowed to decay to nothing.

To use the program, you need to make sure that your disk has at least 100k of free space on it. You then need to create a directory in which to store the wavetable it creates. To do this, enter the command `*CDIR WaveTables`. You should then run the program and when it asks you for a name for the wavetable, type in the name "Brass". It will then store the wavetable it has created with the name Brass in the WaveTable directory.

Using The New Voice

To load the Brass wavetable as a new voice you have to enter the following commands (assuming you are at the BASIC prompt):-

```
*DIR WaveTables
SYS"OS_Module",14,"WaveSynth%Brass"
QUIT
*RMREINIT WAVESYNTH BRASS
```

(N.B. – the case of the characters must be adhered to in the second line.)

If you now change to the Welcome Disc (type `*MOUNT` to cause Arthur to recognise the changed disc and read its catalogue, but **do not** press <ctrl-break> to reset the machine), and type `*DESK`, you can move to the Music Editor program.

When the Music Editor has loaded, select `SETUP` from the main menu and then `INSTRUMENT` from the `SETUP` menu. If you now cycle through the available voices by pressing the `SELECT` button on the mouse, you will see an extra voice is now available called "WaveSynth-Brass"! If you load in `TUNE1` or `TUNE2`, or one of your own, you can then listen to it. It should sound at least a little reminiscent of a member of the Brass family!

In the next article I will explain some of the extra facilities which the developed version of the BASIC program below can offer, such as additive harmonic synthesis and complex FM synthesis. This month's program uses simple FM synthesis!

Support From Acorn

In concluding the article I must make clear Acorn's stance with respect to WaveSynth and our use of it. This is a totally unsupported feature of the operating system, the capabilities of the wavetable system may change (with later versions of Arthur), and Acorn will enter into no correspondence about any aspects of this article or about WaveSynth.

Acorn admit that this might seem a bit harsh, but they argue that the sound system is very complex and they cannot spend a lot of time answering technical queries about this unsupported (and undocumented) aspect of it.

References

Chamberlin Hal, "A Sampling of Techniques for Computer Performance of Music" in "The Byte Book of Computer Music", Byte Books, 1979 (ISBN 0-931718-11-2)

Chamberlin Hal, "Advanced Real-Time Music Synthesis Techniques", BYTE April 1980

Roads Curtis and Strawn John (editors), "Foundations of Computer Music", The MIT Press, 1987 (paperback edition - ISBN 0-262-68051-3)

Tait Simon, "Music Micro Please", Personal Computer World, Dec 1983 - Feb 1984 (3 articles)

```

10 REM > FMBrass14
20 REM
30 REM Build Segmented Wavetable,
   and the correct header block(s)
40 REM
50 REM © Acorn Computers, 1988
60 :
70 MODE 0:PRINT "FM Wavetable
   Voice builder..."
80 :
90 NBkPts% =1000
100 :
110 DIM AmpEnd(NBkPts%-1),AmpTime
   (NBkPts%-1)
120 DIM ModEnd(NBkPts%-1),ModTime
   (NBkPts%-1)
130 :
140 REM amp breakpoints in centisecs
150 DATA 0,1.0, 1,1.0, 30,0.8, 300,
   0.8, 330,0, 0,0
160 :
170 REM mod breakpoints

```

```

180 DATA 0,3.0, 5,7.0, 30,4.5, 300,
   6.5, 330,0, 0,0
190 :
200 REM carrier/modulator frequency
   ratios
210 DATA 1.0, 1.001
220 :
230 bp%=0
240 READ t,v
250 AmpTime(bp%)=FNTime(t)
260 AmpEnd(bp%)=v
270 REPEAT
280   READ t,v
290   t = FNTime(t)
300   IF t>AmpTime(bp%) THEN
310     bp%+=1
320     AmpTime(bp%)=t
330     AmpEnd(bp%)=v
340   ENDIF
350 UNTIL t < AmpTime(bp%)
360 AmpBP% = bp%
370 EndTime = AmpTime(bp%)
380 :
390 bp%=0
400 READ t,v
410 ModTime(bp%)=FNTime(t)
420 ModEnd(bp%)=v
430 REPEAT
440   READ t,v
450   t = FNTime(t)
460   IF t>ModTime(bp%) THEN
470     bp%+=1
480     ModTime(bp%)=t
490     ModEnd(bp%)=v
500   ENDIF
510 UNTIL t < ModTime(bp%)
520 ModBP% = bp%
530 :
540 READ FC, FM
550 :
560 nWaves%=EndTime DIV 256
570 :
580 PRINT nWaves%;" wavetable
   segments maximum"
590 :
600 WaveSize%=nWaves%
610 WaveHeader%=((64 + (nWaves%+1)
   *8)+255)DIV256
620 :
630 BuffSize%=(WaveHeader%
   +WaveSize%)*256
640 DIM Buff% BuffSize%-1

```


Sound Synthesis

```

650 WaveTable% = Buff%+WaveHeader%
                                *256
660 WaveDescriptor% = Buff%+64
670 descriptor%=8
680 :
730 ampBP%=0
740 modBP%=0
750 t%=0
770 endseg%=0
780 WHILE t% < EndTime
790   IF t%>=AmpTime(ampBP%) THEN
800     A=AmpEnd(ampBP%)
810     AT=AmpTime(ampBP%)
820     ampBP%+=1
830     dA=(AmpEnd(ampBP%)-A)/
        (AmpTime(ampBP%)-AT)
850   ENDIF
860   a=A+dA*(t%-AT)
870   IF t%>=ModTime(modBP%) THEN
880     I=ModEnd(modBP%)
890     IT=ModTime(modBP%)
900     modBP%+=1
910     dI=(ModEnd(modBP%)-I)/
        (ModTime(modBP%)-IT)
930   ENDIF
940   i=I+dI*(t%-IT)
950   IF (t%MOD256)=0 THEN
960     PRINTTAB(0,2); "Segment: ";
        t%DIV256
970     WaveDescriptor%!0=&FF+(51
        <<9):REM one waveform
980     WaveDescriptor%!4=((
        descriptor%+1)<<16)
        +WaveHeader%+(t%DIV256)
990     WaveDescriptor%+=8
1000    descriptor%+=1
1010    IF AmpEnd(ampBP%)=0 AND
        endseg%=0 THEN endseg%=t%DIV256
1150  ENDIF
1160  p=2*((t% MOD 256)+0.5)*PI/256
1170  x=a*SIN(FC*p + i*SIN(FM*p))
1180  l%=FNDAClog(x)
1200  WaveTable%?t%=l%
1210  t%+=1
1220 ENDWHILE
1230 :
1260 PRINT "EndSeg (gate-off)
        segment: ";endseg%
1270 WaveDescriptor%!0=0
        :REM end waveform
1280 WaveDescriptor%!4=0
1290 :

1300 PROCHeader
1310 :
1320 PROCSave
1330 END
1340 :
1520 DEF PROCHeader
1530 REPEAT
1540   PRINT "Name";
1550   INPUT name$
1560 UNTIL LENname$>0 AND LENname$<12
1570 FOR A%=0 TO 63 STEP 4
1580   Buff%!A%=0
1590 NEXT
1600 title$="!WT:"+name$
1610 $Buff%=title$
1620 Buff%?(LENTITLE$)=0
1630 Buff%!16=BuffSize%
1640 Buff%!20=8
1650 Buff%!24=8
1660 Buff%!28=8
1670 Buff%!32=8
1680 Buff%!36=8
1690 Buff%!40=8
1700 Buff%!44=8
1710 Buff%!48=8
1720 Buff%!52=8+endseg%
1730
1740 ENDPROC
1750 DEF PROCSave
1760 OSCLI("SAVE WaveTables."
+name$+" "+STR$~Buff%+
" + "+STR$~BuffSize%+" 0 FFFFFFFD00")
1770 OSCLI("STAMP WaveTables."+name$)
1780 ENDPROC
1810 :
1820 DEF FNTIME(t)=t*256:REM convert
        centisecs to sample periods
1830 :
1840 REM log of N in the range +/- 1
1850 DEF FNDACLOG(N)
1860 LOCAL S%,N%:IF N<0 THEN S%=&01:
        N=ABS(N) ELSE S%=&00
1870 IF N=0 THEN = S%
1880 N%=0.5+INT(N*4096+16)
1890 IF N% > &FFF THEN = S% OR &FE
1900 WHILE N%>=32
1910   N%=N%>>1 : S%=S%&20
1920 ENDWHILE
1930 =S% OR ((N%AND&F)*2) A

```

Playing Tunes in BASIC

Joc Carlton

The sound system on the Archimedes is somewhat more complicated than on the Beeb. BASIC programs to play simple tunes on the Beeb will not work when transferred to the Archimedes and the information in the User Guide and the Reference Manual may well be correct, but it isn't very easy to understand! With this in mind I set out to write a simple BASIC program to enable tunes to be played from DATA statements. To do this it is necessary to understand TEMPO, BEATS and VOICES.

VOICES

To take the simplest of these commands first; you can have 1, 2, 4 or 8 separate sound channels open at any time and to select this number you simply enter VOICES 1, or VOICES 2, etc. If you use VOICES 3, four channels are opened. VOICES 5, 6 or 7 opens eight channels. It is necessary to assign a "voice" to each channel used and to see what voices are available just type *VOICES. To assign the voice to a channel, type *CHANNELVOICE <channel> <voice>; for instance *CHANNELVOICE 1 2 means that all SOUND calls to channel 1 will use the StringLib-Soft sound envelope.

TEMPO and BEATS

Tnex two commands are a little more complicated and are to some extent dependent on each other. TEMPO sets the "speed" of the music. The default value is &1000, which sets the length of a beat to one hundredths of a second (centi-seconds). It is best at this stage, to accept the default value and concentrate on the BEATS statement, which sets the number of beats in a bar – not musical beats, but TEMPO beats. The shortest note in normal music notation is the demisemiquaver, so we need to set a value for this. Many tunes include triplets, that is groups of 3 notes to be played in the normal time for 2, shown on music scores as 3 notes with a tie-bar and the figure 3 over them. Many scores also show dotted notes, which are played one and a half times as long as the standard note. So, the value of our note must be such that we can show the whole or half or

two-thirds of the value and the smallest value that fits these criterion is 6. Multiplying this up for notes of other values, we get the following:

Name	Note	Dotted	Triplet
Demisemiquaver	6	9	4
Semiquaver	12	18	8
Quaver	24	36	16
Crotchet	48	72	32
Minim	96	144	64
Semibreve	192	288	128

SOUND

Having decided the length of our notes, we now have to play them and this is where the SOUND statement comes in. On the Archimedes, SOUND has five parameters: channel, volume, pitch and duration (all the same as on the Beeb) and a new parameter "after".

Channel can be any number between 1 & 8 and these numbers relate to the channels opened by the VOICES statement.

Volume carries a value between 0 (mute) and -15 (maximum).

Pitch is a number between 0 & 255 and there is a table in the User Guide, page 183 to relate this number to musical notes.

Duration is measured in twentieths of a second, so at the default setting of TEMPO, 1 corresponds to 5 beats (5 centi-seconds). Actually, this parameter does not seem to be critical, as the Archimedes cuts off a note when it is ready to play the next one, so as long as you make sure it's long enough, it doesn't matter what value you put against this, unless you are trying for a staccato effect by clipping the note off before the next is due.

After counts the beats from the beginning of the bar and starts to play the note on the specified count. For example, if we are in common time (4/4) and BEATS is set at 192, the four (musical) beats would fall on beats number 0, 48, 96 & 144. As BEATS reaches 192, it resets to zero so the next bar can begin. Having set the 'after' parameter for one

channel, it is possible to synchronise the notes played in the other channels by entering -1 on the other SOUND commands, although this is really useful only if all notes are to be synchronised.

NewWorld Program

The program, "NewWorld", plays a 3-voice version of the theme from Dvorak's New World Symphony. There are two additional commands connected with sound used in this program - *SPEAKER OFF at line 55, which switches off the internal speaker because I play the Archimedes' sound through my hi-fi. Remove this line if you want to use the internal speaker, or you won't hear very much! Lines 110, 130 & 150 use the STEREO command to spread the voices across the "stage", but you will only hear the effect of this through an external system such as the mini-headphones from a personal stereo.

Playing Tunes

PROctune_one, PROctune_two etc form the active part of the program, reading the data and then playing the notes. The format of the data is the number of notes to be played in the particular bar for voice 1, followed by the pitch, duration and "after" parameters for each note. This is then repeated for voices 2 & 3. Each line of data is for one complete bar. Line 790 waits for the value of BEATS to reach zero (the start of a bar) and line 820 checks that the bar is under way before sounding a note. Such is the Archimedes' speed that this line really is essential! Because we have built in this delay, it is prudent to add 10 to each "after" parameter, so the common time beats fall on BEATS 10, 58, 106 & 154.

If you want to SEE just how fast the Archimedes handles the sound commands, change line 60 to MODE 15, add line 795 CLS and add lines 835, 885 and 935, each as *GCOL RND(63):ELLIPSEFILL RND(1279), RND(1023), RND(200)+50, RND(200)+50, RND(1)*.

Notes about LOCAL DATA statements

Adrian and Ed. modified Joc's program to use the technique for putting DATA statements into procedures that we mentioned in the last issue. This technique has been further refined (i.e. it works

properly now!) but may need a little explanation. When PROctune_one is executed, it sets up the local error handler and reaches line 310 where and error is deliberately generated. It then executes line 300: RESTORE ERL sets the DATA pointer to line 310 where the error occurred, RESTORE ERROR (a different use of the same keyword) restores the error handler to what it was previously, then the data is used by PROCplay. If you would prefer to have the procedure more self-contained rather than making a call to a procedure from the error-handler, you could use:

```
DEF PROCtune_one
LOCAL flag
LOCAL ERROR
flag=TRUE
ON ERROR LOCAL RESTORE ERL
IF flag% THEN flag=FALSE:ERROR
RESTORE ERROR
PROCplay
DATA 3,65,14,10, etc...
DATA
ENDPROC
```

```
10 REM > $.NewWorld
20 REM by Joc Carlton
30 REM March 1988
40 REM Mods by Paul & Adrian
50
55 *SPEAKER OFF
60 MODE 0
70 TEMPO &1000
80 BEATS 192
90 VOICES 3
100 *CHANNELVOICE 1 3
110 STEREO 1,0
120 *CHANNELVOICE 2 2
130 STEREO 2,100
140 *CHANNELVOICE 3 2
150 STEREO 3,-100
160 FOR Z%=1 TO 2
170   PROctune_one
180   PROctune_two
190   PROctune_three
200   PROctune_three
210   PROctune_one
220   PROctune_four
230 NEXT
240 END
250
260 REM Data for New World by Dvorak
270
```



```

280 DEF PROCtune_one
290 LOCAL ERROR
300 ON ERROR LOCAL RESTORE ERL:
    RESTORE ERROR:PROCplay:ENDPROC
310 ERROR
320 DATA 3,65,14,10,81,4,82,81,19,106
    ,2,33,19,10,21,19,106,1,5,38,10
330 DATA 3,65,14,10,61,4,82,53,19,106
    ,2,33,19,10,21,19,106,1,5,38,10
340 DATA 4,61,14,10,65,4,82,81,14,106
    ,65,4,178,1,33,38,10,1,1,38,10
350 DATA 1,61,38,10,2,33,19,10,25,19
    ,106,1,1,38,10,99
360 ENDPROC
370
380 DEF PROCtune_two
390 LOCAL ERROR
400 ON ERROR LOCAL RESTORE ERL:
    RESTORE ERROR:PROCplay:ENDPROC
410 ERROR
420 DATA 3,65,14,10,81,4,82,81,19,106
    ,2,33,19,10,21,19,106,1,5,38,10
430 DATA 3,65,14,10,61,4,82,53,19,106
    ,1,37,38,10,1,21,38,10
440 DATA 4,61,14,10,65,4,82,61,14,106
    ,53,4,178,2,41,19,10,49,14
    ,106,2,25,19,10,33,19,106
450 DATA 1,53,38,10,1,33,19,10,2,21
    ,19,10,5,19,106,99
460 ENDPROC
470
480 DEF PROCtune_three
490 LOCAL ERROR
500 ON ERROR LOCAL RESTORE ERL:
    RESTORE ERROR:PROCplay:ENDPROC
510 ERROR
520 DATA 3,89,14,10,101,4,82,101,19,
    106,1,53,38,10,2,25,19,10
    ,41,19,106
530 DATA 3,97,9,10,81,9,58,89,19,
    106,1,53,38,10,2,33,19,10
    ,25,19,106
540 DATA 4,89,9,10,101,9,58,97,9,106
    ,81,9,154,1,53,38,10,2,25,19,10
    ,33,19,106
550 DATA 1,89,38,10,1,53,38,10,1,25
    ,38,10,99
560 ENDPROC
570
580 DEF PROCtune_four
590 LOCAL ERROR
600 ON ERROR LOCAL RESTORE ERL:
    RESTORE ERROR:PROCplay:ENDPROC
610 ERROR
620 DATA 3,65,14,10,81,4,82,81,19,106
    ,2,33,19,10,61,19,106,2,5,19,
    10,49,19,106
630 DATA 3,101,14,10,109,4,82,113,19
    ,106,2,53,19,10,53,19,106,2
    ,41,19,10,33,19,106
640 DATA 4,109,14,10,101,4,82,109,9
    ,106,89,9,154,1,41,38,10,1
    ,25,38,10
650 DATA 1,101,38,10,2,33,19,10,53,19
    ,106,1,21,19,10
660 DATA 1,89,38,10,1,73,38,10,4,61
    ,14,10,53,4,82,61,9,106
    ,41,9,154
670 DATA 1,81,38,10,1,65,38,10,1,53
    ,38,10
680 DATA 1,81,38,10,1,65,38,10,1,5
    ,38,10,99
690 ENDPROC
700
710 DEF PROCplay
720 REPEAT
730 READ N1
740 IF N1<>99 THEN PROCplayit
750 UNTIL N1=99
760 ENDPROC
770
780 DEF PROCplayit
790 REPEAT UNTIL BEAT=0
800 FOR X%=1 TO N1
810 READ P,D,A
820 REPEAT UNTIL BEAT<>0
830 SOUND 1,-15,P,D,A
840 NEXT
850 READ N2
860 FOR X%=1 TO N2
870 READ P2,D2,A2
880 SOUND 2,-15,P2,D2,A2
890 NEXT
900 READ N3
910 FOR X%=1 TO N3
920 READ P3,D3,A3
930 SOUND 3,-15,P3,D3,A3
940 NEXT
950 ENDPROC A

```

Errata

Programmers' Reference Manual, page 283
 "Types don't match" is error number &C5 (197) not
 &C6 (198)

User Guide Issue 3, page 313 the result of doing an
 OPENUP when the file already exists is NOT to
 delete the old file, nor does it create a new one! It
 simply opens it for up-dating. **A**

EMR SoundSynth Review

Kevin Smith & Matthew Treagus

The EMR SoundSynth package forms only a small part of a complete music system for the Archimedes called "Arpeggio" of which I am sure we will be hearing more in the near future. It is just as well that the entire package is coming out as different modules because it will make it much more attractive to users who only need certain facilities.

The SoundSynth is essentially a sound voice generator which can be linked to the rest of the Arpeggio system and to BASIC via Arthur (the OS) using the EMR Waveform Filing System. The program itself is very neat and tidy and boots up into a clever but completely useless title screen featuring a spinning EMR logo. You then progress on to the Arpeggio Music System Main Menu which is the focal point of the system providing links to the SoundSynth and 16 other modules including the Editor, Performer, Sampler and Midibank.

The SoundSynth menu screen is set out with the current waveform in the top left hand corner – each voice being made up of as many as 512 different waveforms. Just below this is a linear representation of the sound showing loops, length and position of the current waveform in the sound. The top right hand side contains general information and the bottom half of the screen contains the thirty options blocks and the help window (if selected) These thirty options are explained briefly below.

Simple Commands

The first set of options that we shall deal with are the simpler ones which do not need much explaining.

Help On/Off – Help on produces a help window on the screen.

Sample – Allows you to obtain a sound wave from the A448 sound sampling board, also available from EMR.

Load – Loads a sound into memory.

Save – Saves a sound to disk.

Append – Joins a sound loaded from disk on to the current sound in memory.

Show – Displays a 3-D graphical representation of the complete sound.

Hear – Allows you to hear the sound by pressing keys on a simple keyboard layout. This is just meant as a testing facility.

Master Volume – Controls the overall volume of the sound output.

Print – This provides access to two sub options :- 'Sample' gives a dump of the sound in a two dimensional form. The sound can be very long, so the user is provided with a method of scaling the printout in order to prevent the printer from throwing seven feet of paper all over the desk! '3D', gives a screen dump of the graphic image displayed by the 'Show' option.

Editing Commands

The next set of options are concerned with editing the sound directly. References to 'waves' can mean all or part of the sound. Each block of waves is selected by using the mouse to select the limits of the block. The keyboard is only used to enter filenames or echo delays.

Move – Moves a wave from one position to another.

Insert Area – Allows you to insert a blank area into the current wave.

Insert – Inserts waveforms directly into the sound, increasing the length of the sound as more waveforms are added.

Copy – This copies waves of the sound from one position to another.

Delete Area – Deletes waves inside the currently displayed waveform.

Delete – Deletes waves and compacts the remaining waves together.

Wipe – Deletes complete sets of waveforms from the sound and leaves the area deleted as blank space rather than reducing the overall size of the sound.

Edit – Edits the currently displayed waveform by

moving the mouse pointer inside the waveform window.

Harmonics – Edits individual waveforms by changing amplitudes of different harmonic frequencies. These varying frequencies combine to form a clear sound which is played as you edit. To do this, a special window is displayed showing the various amplitudes in the form of a bar chart, and the mouse pointer can be used to pull the bars up or down on the graph.

Sound Enhancement

The next set of options are slightly more complicated as they provide the musically inclined user with some very ingenious facilities to enhance their sound.

Alter With Equation – Allows you to change sections of the current sound with a mathematical equation. It accepts all the mathematical functions from BASIC and has some added special features.

Gate On – This defines the amplitude envelope of the sound while the sound is triggered.

Gate Off – This defines the amplitude envelope of the sound after the sound trigger has been released.

Interpolate – Gradually changes one waveform to another by creating a set of intermediate waves.

Random – This option generates random waves in the sound, producing white noise.

Echo – This adds echo waves on to the sound. You can set the echo volume and the delay period between the main sound and the echo.

Reverse – Physically reverses part or all of the sound.

Overlay – Merges the current sound in memory with one loaded in from disk.

Scale – Makes the amplitude of the sound smaller or larger.

Slapback – Repeats waves after the main sound has been played. This is similar to echo.

Looping – Replays part or all of sound as if it were part of the original sound.

Comments and Criticisms

As a stand alone package it is only really useful as a voice generator for other programs. It makes voice

generating very easy using well presented screens and good graphical representations of the sounds.

Overall, this package is excellent. It is well presented and easy to use, once you have got to grips with all the various jargon terms (see glossary).

There are a few minor criticisms that we have...

The first is the documentation (a twenty page manual) which is only just adequate, although the on-screen help and the generally user-friendly layout makes up for part of this. The Waveform Filing System is well documented.

One small problem when actually using the package was that mouse controls can effect parts of the screen which are not currently under the mouse pointer. (Confusing, but not that important)

One feature that we would have liked to have seen is the ability to change the waveform number whilst within the edit or harmonic options, as opposed to having to leave the option first in order to change the waveform number and then re-entering the option.

Another feature that was missing was an 'undo' feature on the editing options which would provide a way of restoring the previous waveform after making a mistake. This can be quite annoying.

Interaction

Interaction between the package and BASIC is made easy by the 'Waveform Filing System' module. The 'SoundSynth' has the ability to interact with future parts of the 'Arpeggio' suite by means of the initial main menu.

Conclusion

At £50.00 this package is a very reasonable buy for those who wish to get the best out of their Archimedes sound chip and for those who wish to expand and buy the rest of the 'Arpeggio' suite or the A448 sound sampling board.

Note

EMR are currently offering a special promotional offer of the package for £40.00, but this is a limited offer. EMR also produce library discs of sound that can be used in the Waveform Filing System and in the SoundSynth. **A**

Languages Corner

Brian Cowan

This will hopefully be the start of a regular feature about using different languages on the Archimedes. Brian has very kindly agreed to start us off by editing the first lot of material but other folk have offered help and they may well take it in turns to do this section of Archive. If anyone has any questions or hints and tips, please write in and we will do our best to deal with them each month.

Books

"Computer Languages: A Guide For The Perplexed" by Naomi S. Baron, published by Penguin at £9.95 (ISBN 0-14-02.2807-1)

This is a delightful book about the whole range of computer languages. It is equally suitable for the novice who knows nothing and the experienced user who wants to know about other languages. The first part of the book discusses general aspects of computer languages from a linguistic point of view. This is followed in the second part by individual sections on twenty two languages from Ada, through BASIC, C, Modula-2, FORTH, FORTRAN, LISP, Pascal etc to SNOBOL.

Each section gives a tabulated profile of the language which gives an explanation of its name, a discussion of the language structure, its functions, its genealogy presented usually as a family tree, its history and a summary of its versions/dialects. There is also mention of special characteristics. It then expands on these points, discussing the history, often including political and "big business" considerations. Some sample program segments are usually given and the sections conclude with speculation about the future of the language.

The section on BASIC is very fair considering the unpopular press this popular language usually gets from computer experts! This section is also distinguished by the absence of any mention of BBC BASIC! This may be explained by the fact that the author of the book is American; she admits that the contents of the book represents her own

experience. Each section ends with a well documented list of references which includes books and magazine articles.

Finally there is an appendix which discusses briefly some seventeen other languages, among which are COMAL, Occam, POP and Turing.

This is a well written book and it is an easy read. It is ideal for anyone in the process of choosing a language. For the seasoned programmer it gives a superb overview of the gamut of computer languages.

C Programming – Apparently Acorn recommend their programmers to read **"C: A Reference Manual"** (Second Edition) by Harbison and Steele, published by Prentice Hall, ISBN 0-13-109802-0.

All serious C users, at some stage you will want to look at the book by the author of the language, Dennis Ritchie, **"The C Programming Language"** by Kernighan and Ritchie, published by Prentice Hall, 1978, ISBN 0-13-110163-3.

Other Languages for Archimedes?

Modula-2 – It seems that Acorn are not going to release their Archimedes version of Modula-2. They have a version which is used within Acorn but it needs 2 Mbytes of RAM to compile and so could not be used on the 300 series machines.

BCPL – There must be an Archimedes BCPL since this is the language in which the ill-fated ArcWriter was written. It appears that Acorn are not going to release this either.

Relocatability

The current versions of the compilers for C, Fortran 77 and Pascal do not produce relocatable code. The problem is in the language libraries. This means that you can not use these compilers for writing code for relocatable modules. Relocatable versions of the languages are being produced and the first to appear should be C towards the end of this year. It should come with user-friendly linker and utilities for producing relocatable modules complete with header code etc.

Parameters in Command Files

Mike "Beard" Williams (*not to be confused with any other Mike Williams you might know!*) writes in reply to Lawson Wakefield's query about command file parameters:

Suppose we often want to do the same series of operations to different data files but don't want to type all the commands in each time. We could write a program in BASIC, using OSCLI commands, but this is not always appropriate. What we can also do is to write a command file in which the name of the data file is a variable. When we want to run it we first SET a value for the variable, then execute the command file.

This simple example will execute a SCREEN LOAD command to a filename contained in the variable SFILE.

```
*BUILD LOADIT
*SETEVAL ALIAS$TMP "SCREENDUMP "
                                     +SFILE
*TMP
*UNSET ALIAS$TMP
```

Explanation: line 1 evaluates the expression "SCREENDUMP "+SFILE and stores the result in a temporary alias called ALIAS\$TMP, line 2 causes ALIAS\$TMP to be executed, line 3 is optional; it leaves things tidy by deleting the temporary alias.

```
*SETTYPE LOADIT &FFE
```

can be used to tell Arthur that this is a command file. When we want to use the command file we type, e.g.

```
*SET SFILE $.USERPICS.MYFILE
*LOADIT
```

This might seem like a lot of work, but there are some languages where the compilation of a program is performed in a complex series of stages which could not be controlled from BASIC using OSCLI, but a command file using variables for the file names could be used.

Floating Points

Those considering the use of a compiled language for number crunching on the Archimedes will, no doubt, have been disappointed in the speed comparisons at the end of last month's review of Archimedes Fortran 77. In certain respects the

figures were a little unfair since the other machines both had floating point coprocessors installed whereas the Archimedes was using the floating point emulator. This is no comfort for 300 series owners whose machines will not accommodate a coprocessor. However there are some modest speed increases in later versions of the emulator and a new memory controller chip MEMC1a speeds up the MUL instruction.

Owners of 400 series machines will be looking forward to the release of the hardware floating point coprocessor system. This is expected towards the end of this year, although the cost will be in the neighbourhood of £700. Floating point programs should run some eight times faster with the coprocessor. Comparison with other machines would then be a very different story! Early results (courtesy of Roger Wilson) comparing the software emulator and hardware floating point coprocessor gave the following benchmark results, (both double precision):

	Emulator	Co-processor
LinPack	11.46 KFlops	92.25 KFlops
WhetStone	88 KWhets	674 KWhets

There is no point in using single precision, since double or extended precision are only minutely slower. Most of the overhead is in working out what the instruction does rather than doing it! The above tests were done using the floating point emulator version 2.50 and using the new MEMC chip, so current machines will not be quite as fast.

So where does this leave the humble BASIC user? If the Acorn 32016 systems are anything to go by, a version of BASIC using the floating point coprocessor was eventually released. Maybe the same will happen for Archimedes BASIC. A BASIC assembler that supports the floating point mnemonics is certainly required, and while we are at it, what about Acorn doing a compiled version of BASIC 5? **A**

Both Gerrald Fitton and John Smith (whom I inadvertently called A.J. Smith last month. Sorry!) have been working independently on floating point assemblers – we have now put them in contact with each other and await developments! Ed.

FORTRAN 77 Graphics Library

Gwyneth Pettit

This FORTRAN library published by CCD Computer Services, price £49.50, consists of a disk containing the Graphics library file (GxLib), a command file for setting up a Hewlett-Packard serial plotter and four demonstration files (FORTRAN source code, sample data and demonstration output) plus an excellent user guide.

The library provides over 90 subroutines which can be called from a FORTRAN source code text file with simple commands such as:

```
CALL GV PLOT(IX, IX, IX)
```

(c.f. PLOT K%, X%, Y%). An object code file is then prepared by including the library file, GxLib, with the main f77 library file at the linker stage.

Subroutines

The subroutines can be divided into two groups:

- a group of 33 calls which access various VDU calls (including those for mode, colour, windows, and plotting filled and outline shapes).
- a group of general calls which generate sophisticated graphs (including 3D data representations), histograms, pie-charts etc.
- Also access to Arthur calls – see "Post Script".

Without using a graphics library, a FORTRAN user could access Archimedes graphics directly through VDU calls, using commands based on:-

```
WRITE CHAR(IX)
```

However this can get complicated for users not already familiar with BBC BASIC VDU commands, particularly with VDU 23 (which requires 9 parameters) or VDU 25 (PLOT) commands, for which screen coordinates have to be sent as high byte-low byte, because they exceed the maximum byte value of 255. The VDU subroutines in this library relieve the programmer of these digital gymnastics and give a smooth 'user friendly' access to the many graphics effects possible through VDU commands. FORTRAN users with experience only of mainframe terminal displays in monochrome will particularly enjoy exploiting the 256-colour modes, which make the Archimedes worth (nearly) every penny.

General calls

The general calls are divided into a number of groups including:

- General calls to initialise the library by setting up default parameters etc. or to set up an output device.
- Routines to define, draw, scale, mark and label axes (x-, y- and/or z-axes) and to plot data pairs.
- Routines to scale and output characters or strings, including the opportunity to output at any angle of slant or rotation and to prepare and output user-defined characters.
- Routines for different line styles
- Routines for several standard statistical presentations, e.g. pie-charts, histograms etc.

These routines are specifically designed for programmers wishing to display calculated data, using a variety of graphical representations. They avoid a considerable amount of tedious 'house-keeping' work such as setting up axes and divisions, the scaling of parameters and the organization of text output. Thus, graphs of mathematical functions can be displayed for scientists and engineers, and a wide range of business graphics can be programmed without the need to understand Archimedes graphics beyond the fundamentals of screen modes and pixel counts.

The routines do not have facilities for scaling, windowing and clipping, nor are there facilities for picking and dragging shapes from menus (which could exploit GCOL 3 for EOR or Exclusive OR drawing, invaluable for CAD users). The term 'graphics library' is therefore in a somewhat limited sense, but what it does, it does well.

Output can be directed to the screen or a file. Files can then be output to a number of output devices, including Hewlett Packard plotters, for which the necessary software is included. Alternatively the output could be sent directly to a Plotmate plotter, which responds directly to Archimedes VDU commands, if the Plotmate can be interfaced with the Archimedes. The reviewers were not able to test this facility.

Documentation

The manual is particularly helpful, filling in much useful information omitted from the Acorn FORTRAN manual. It gives guidance on the recommended directory structure of working disks, using either a single or double disk drive and, most usefully, a suggested command file to create a good working environment and maximise the memory available. This includes suggestions for a number of alias\$ and RMKILL commands and for loading the floating point emulator. It does not give advice for those of us fortunate enough to own a 440 with a hard disk but this is straightforward in practice. The 440 has effectively no memory limitations and you can load the GxLib FORTRAN library file in the same directory on the hard disk as the f77 FORTRAN library file.

Summary

As a library of FORTRAN output routines this Graphics Extension Library is well prepared and well documented, making it easy to use and saving a lot of programming time, which can easily offset the initial cost of the library. By writing a small FORTRAN program to change mode, transmit VDU5, to move the graphics cursor and print a message, and to draw a filled triangle, I was able to feel that even in FORTRAN my Archimedes was comfortably familiar to a BBC user; those who have come straight to the Archimedes without using the BBC B or Master will find this graphics library extremely flexible and comprehensive compared with many other micros (certainly compared with most minis and mainframes).

P.S....

As this review was being prepared, a very welcome extension to it arrived in the post, providing access to the Archimedes operating system, thus giving the FORTRAN programmer a rare ability to control graphics programs interactively. There is an INKEY function and access from FORTRAN to machine code routines, SWI calls and "" (OSBYTE) commands and much, much more. This will make it possible to use the mouse and even better, the more ergonomic graphics input devices, to access the WIMP environment and to perform*

simple and useful functions such as cataloguing a disk or clearing buffers. This is a major extension to an already impressive FORTRAN support library.

*Purely for output, however, this first version of the Graphics Extension Library is very useful and can be recommended as an almost necessary addition to the Acorn FORTRAN disk, since it adds commands similar to those provided by GINO or similar FORTRAN support packages on mainframes and minicomputers, at a fraction of the combined cost. We shall have more information about the cost of the second stage in the next review, but there seems no doubt that the whole system will be more than good value. **A***

Bug or Feature?

- ***FX220,n** allows the selection of a key other than the escape key to generate the escape event. For example, ***FX220,1** makes <ctrl-A> generate an escape. Unfortunately, if you use ***FX220,0**, commonly used on the BBC micro, you will find that not only does <ctrl-@> generate an escape, but also the pound sign and grave accent!

This was pointed out to us by Contex Computing who had this problem with their Bank Manager program. If you have version 3.0 or 3.1 of their program, send your diskette back to them and they will replace it.

- One reader says he gets the "Unknown IRQ at &00000000" error even with no RS423 lead attached – any ideas, anyone?

Now some answers from Acorn...

- The problem of the **pointer breaking up** is due to the nature of the parameters given to VIDC: it is ameliorated in future versions of Arthur. In the mean time, declare pointer bit maps with the active area at the left. (or do they mean right?)

- **TRACE in BASIC** gives very little information simply because BASIC doesn't execute many of the lines one would naively expect – which is why it goes so quickly! However, BASIC V version 1.03 has a sophisticated alteration to slow it down (i.e. execute more of the lines one might think it was going to) when TRACE is active! **A**

MS-DOS Column

Ken Biddle

This will hopefully be the start of a regular feature about using PC software on the Archimedes. Ken has very kindly agreed to start us off by editing the first lot of material but other folk have offered to help and they may well take it in turns to do this section of Archive. If anyone has any questions or hints and tips, please write in and we will do our best to deal with them month by month.

Ken writes... I am starting a list this month of software that **does** run using the PC-emulator. If you can add to the list in any way then please write in to me.

The documentation that is distributed with the PC-emulator is virtually non existent so we will be providing more information each month on the DOS environment and its commands and also GW-BASIC which is also supplied with the package.

PC-Emulator Software List

Wordstar Release 3
Wordstar Release 4
Microsoft QuickBASIC
Norton Utilities
Dbase III
Zortech C Release 2
SideKick Release 1.56A
DOS Release 3.3

Hints and Tips

Allocating more RAM to DOS

The Archimedes RAM can be allocated in various ways and it is important when using the PC-emulator that as much RAM as possible is free for the Simulated DOS environment.

The following is one way of achieving the maximum amount of RAM for DOS.

Backup your existing PC-emulator disk and then put it away in a safe place. It is recommended that you should now only use your newly created disk as a working version of the emulator. You should now

create 2 new files called !START and !END in the following way :

```
*BUILD !START
*CONFIG.SCREENSIZ 10
*CONFIG.FONTSIZE 0
*CONFIG.SPRITESIZE 0
*CONFIG.RMASIZE 0
*RMKILL FONTMANAGER
*RMKILL STRINGLIB
*RMKILL PERCUSSION
*RMKILL ARMBASICEDITOR
*RMKILL WINDOWMANAGER
*RMCLEAR
*RMTIDY
```

Now press <escape>

```
*BUILD !END
*CONFIG.SCREENSIZ 20
*CONFIG.FONTSIZE 6
*CONFIG.SPRITESIZE 2
*CONFIG.RMASIZE 2
*RMREINIT FONTMANAGER
*RMREINIT STRINGLIB
*RMREINIT PERCUSSION
*RMREINIT ARMBASICEDITOR
*RMREINIT WINDOWMANAGER
```

Now press <escape>

You now have two new files on the disk which should be used in the following way to optimise the amount of RAM available to the DOS environment.

Before you boot up the PC-emulator you should enter the following command - *EXEC !START and then boot the PC-emulator disk as normal.

This will configure the RAM allocations in favour of DOS and will also unplug and resident modules that DOS does not need. Boot up DOS in the normal way and you should find that you have a DOS system of well over 500k

When you have finished in PC-mode and you return to the Arthur operating system, you should type the following : *EXEC !END

Your RAM allocation will now be reset reasonable values and the Modules that were previously disabled are now reinstated.

RAM Disks

It is quite a simple procedure to set up a ram disk and is achieved in the following way:

Type the following :

```
EDLIN CONFIG.SYS
```

```
I
```

```
SYSTEM=RAMDRIVE.SYS xxx 512 64
```

Now press <ctrl-Z>

(xxx is the size of the ramdisk (i.e. 100 would give you a 100k ramdisk.) DOS can now be re-booted <ctrl/alt/del> and a Ramdrive of the specified size will be created.

If you only have one physical disk drive then this ramdisk is extremely useful for transferring programs and or files from one disk to another.

GETFILE and PUTFILE

Finally this month, we will have a quick look at the utilities GETFILE and PUTFILE. These utilities may be used to move programs and or files between the DOS filing system and Acorn's ADFS.

For example:

```
A>GETFILE :0.test B:ken.txt
```

copies the file test from the ADFS format in drive 0 to the DOS format in drive B while at the same time renaming it to ken.txt

```
A>PUTFILE A:ken.txt :0.test
```

copies the file ken.txt from DOS in drive A: to ADFS in drive 0.

Well that's all for this month. Please write in and tell me if you have any specific queries and I will do my best to address them. **A**

Auto-Configuration Re-Visited

Andy Currie

This program is a development of Adrian Look's two programs on automatic re-configuration in Archive volume 1, issue 5.

The idea of this version is that it will not only configure the machine to the requirements of the current application, but it will re-configure the CMOS ram to the settings immediately prior to running it, ready to reset after using the application.

This adaptation takes advantage of the fact that any changes to the configuration of the machine are only acted upon at the next <break>. On the first pass through the program the current settings of the CMOS Ram are saved in the user area of CMOS ram, the memory configuration is reset, and the machine code software reset routine is assembled. The machine is then configured to auto-boot and the software reset called.

On the second pass through the program the original CMOS settings are restored, any unwanted modules are RMKilled using XOS_MODULE so

that inactive modules do not report errors. Key 0 is then loaded with the calling action for the application and the keyboard buffer loaded with the code for Key 0. This is done because when we use *RMTIDY it will switch BASIC out and wipe out our program.

The two lines ENDIF and END are never reached by the program when it runs, but are included as good programming technique.

This particular !BOOT file is for use with the PC_Emulator and will give 520k working space within the DOS environment. The disc should be set with *OPT 4,2 to run !BOOT as a BASIC program.

```
10 REM                >!boot
20 REM Program      Auto Configure
30 REM Version      1.10
40 REM Author       Andy Currie
50 REM Date         2 April 1987
60 REM Program expanded from the
                        original idea by
70 REM Adrian Philip Look in
                        Archive Vol.1 Iss.05
```

Auto-Configuration

```

80
90 *FX 200,2
100
110 DIM code% &FF
120 SYS "OS_Byte",161,16 TO ,,
                                misc flags
130 auto_boot=(misc_flags AND
                                %10000)
140 IF auto_boot=0 THEN
150   PROCsavecmos
160   PROCconfigure
170   PROCassemble
180   *configure Boot
190   CALL code%
200   ELSE
210   PROCrestorecmos
220   PROCkillmodules
230   *KEY0 "PC.Emulate|M"
240   *FX138,0,128
250   *configure NoBoot
260   *RMTIDY
270 ENDIF
280 END
290
300 DEFPROCsavecmos
310 SYS "OS_Byte",161,134 TO
                                ,,default
320 SYS "OS_Byte",162,30,default
330 FOR A%=1 TO 5
340   SYS "OS_Byte",161,A%+142 TO
                                ,,default
350   SYS "OS_Byte",162,A%+30
                                ,default
360 NEXT
370 ENDPROC
380
390 DEFPROCconfigure
400 *configure ScreenSize 10
410 *configure RMASize 0
420 *configure SpriteSize 0
430 *configure RamFsSize 0
440 *configure SystemSize 0
450 *configure FontSize 0
460 ENDPROC
470
480 DEFPROCassemble
490 FOR opt=0 TO 2 STEP 2
500   P%=code%
510   [OPT opt
520   MOV R0,#&3800000
530   LDR R1,[R0]
540   STR R1,[R1,-R1]
550   SWI &16
560   TEQP PC,#&FC000003
570   MOV PC,#0
580 ]
590 NEXT opt
600 ENDPROC
610
620 DEFPROCrestorecmos
630 SYS "OS_Byte",161,30 TO ,,
                                default
640 SYS "OS_Byte",162,134,default
650 FOR A%=1 TO 5
660   SYS "OS_Byte",161,A%+30 TO
                                ,,default
670   SYS "OS_Byte",162,A%+142
                                ,default
680 NEXT
690 ENDPROC
700 :
710 DEFPROCkillmodules
720 SYS "XOS_Module",4,"Debugger"
730 SYS "XOS_Module",4,
                                "SystemDevices"
740 SYS "XOS_Module",4,"Econet"
750 SYS "XOS_Module",4,"Netfs"
760 SYS "XOS_Module",4,"Netprint"
770 SYS "XOS_Module",4,
                                "ARMBasicEditor"
780 SYS "XOS_Module",4,
                                "WindowManager"
790 SYS "XOS_Module",4,
                                "Fontmanager"
800 SYS "XOS_Module",4,
                                "Soundscheduler"
810 SYS "XOS_Module",4,
                                "Stringlib"
820 SYS "XOS_Module",4,
                                "Percussion"
830 ENDPROC A

```


Index of Archive Volume 1: Issues 1 - 6

Based on information provided by Matthew Treagus

+=	6.6	ARMpaint	4.7
-=	6.6	Arthur 1.2	3.3, 5.8, 5.10
1.2 Operating System	2.2, 5.10	Articles, sending to Archive	5.50
2401 colours in mode 15	4.8	Artisan	2.4
3D Button effect	5.26	Versus Arcist	5.9
3D Graphics animation system	6.3	Audio-Visual Publications	1.3
4096 colour screen dump	6.2	Automatic Reconfiguration	5.17
4Mbyte on A300's	2.9		
5-Byte time format	6.28	Back Plane	5.2
5.25" Disk Interface	1.5, 2.7, 3.3, 6.5	BASIC editor	
6502 conversions (RESOURCE)	3.6	as a word pro	4.7
		on Arthur 0.20	3.10
Abracadabra	4.4	BASIC V	
Acorn		+= and -=	6.6
Colour monitors	4.5	Extra help	5.6
Languages	3.5	Help	2.10, 5.6
LISP and Prolog	4.2	Line numbering	2.8
Soft Ltd	1.3	LISTO	2.10
Teletext Adaptor	6.5	More about	6.37
Adaptor SCML Teletext	6.14	OFF/ON	2.10
ADFS		OS Interface	5.2, 6.4
Beginner's Guide	3.40, 4.33	Program Structure	1.28
Bugs	1.6	String handling	4.19
Disc Editor	6.43	Technical Notes	1.32
Hints and Tips	3.45, 5.6	Text files	5.6
Wildcards	6.9	Tips	3.11
APL Interpreter	3.5	Trace problems	6.11
Archimedes		BBC Micro as an I/O Podule	4.6
A400 series	3.3	BBC ROM images in 65Arthur	2.22, 5.7
Index - Programmers' Map	2.4	BBC to Arc - Conversions	5.2
Prices cut	2.2	BEEBUG Ltd	1.3
Tool-Kit Module	1.22	BEEBUG's Masterfile	5.7
Artist	3.6	Boot file for WWPlus	3.11
versus Artisan	5.19	Boot files	6.8
ArcWriter	6.7, 6.9, 6.10	Brainsoft	2.3
ARM		Break/escape effects	4.6
Assembly Programming	2.5	Buzz fix	2.3, 3.7
Second processor	2.4		

Catalogue, reading	5.7	Econet	6.10
Centronics screendump	5.9	EDIT on 0.2 OS	1.9
Clock on desktop	5.7	EDIT ROM from BBC Master	4.7
Clock speed - 4/8 Mhz	2.9	Election 1987	4.4
Colour Blender	3.19, 5.4	Errata	1.2, 2.9
Colour screen dump	6.2	Errors in library procedures	6.9
Colours, lots in mode 15	4.8	Escape effects	4.6
Comms on Archimedes	2.8		
Competition Results	4.46	Fact or Fantasy?	5.50, 6.4
Computer Concepts ROM Podule 3.5, 5.2, 6.3		Fast screenload/save module	5.6, 6.9
CONFIGURE command	2.12	File search utility	5.48
Continuous Processing ROM	3.5	File transfer between WPs	6.8
Control key abbreviations	3.8	File transfer kit	3.2
Copying problems	2.6	Fitting a second drive	3.7
Copying with a single drive	2.10	Floating point emulator	5.3, 6.39
		Font designer	6.2
DATA in library procedures	6.5	Font problems	5.3
Data Switches	6.2	Fortran 77	2.7
Database Management System	6.3	FX 25	6.11
Decrement (-=)	6.6		
Definitions	3.9	Game of Life	6.32
Delete on keypad	3.9	Graphic-Writer	4.2, 6.12
Delta-Base	1.2	Graphics Demo Disc	5.4
Desktop	5.7, 5.8, 6.6, 6.9	Graphics Demo Programs	1.38, 2.31
Games	6.3	Graphics Library, Micro Studio	6.2
Switch off	4.7	GW BASIC	6.9
Desktop manager programs	6.6		
Desktop publishing package	4.4	Hand Pointer	2.41
Diagram II	3.6, 5.23	Hard Discs	5.2, 5.3, 6.26
Diary of an Archimedes User	1.9	Hardcopy module	4.7
Diary		Help	2.10, 5.6
filing	3.9	HELP debugger module	2.30
keys	5.8	HELP information	2.24
problems	6.8	HELP on Modules	2.11
quick use of	5.8	Heraldry program	4.4
DIRCOPY	2.10		
Disabling Modules	6.8	I/O Podule	5.2
Disassembly	5.8	Identifying RamBASIC	6.7
Disk file transfer	1.7	Increment (+=)	6.6
Disk Interface, 5.25"	1.5, 2.7, 3.3, 6.3, 6.5	INTER Series	1.2, 3.5
DIY memory upgrades	2.11	ISO Pascal	4.7

Keys	5.8	Music Computer	1.26
Keys on the numeric keypad	4.5	Naff RTC MONTH	2.9
Library PROC errors	6.9	NAMEDISC – ADFS	5.6
Lid, Removing	2.9	NewsMaster	4.14, 6.16
Life program	6.32	No keyboard present	1.9
Line numbering, BASIC V	2.8	No room in RMA	3.11
Linefeed Problems	6.4	Notepad	6.8
Listings, slow	6.8	Numeric pad keys	4.5
LISTO, BASIC V	2.10	OFF/ON	2.10
LOGO	2.3, 6.2	ON ERROR routine	6.8
LOGOsoft's Master Logo	2.3	Operating System 1.2	2.2, 5.10
Logotron's Archimedes LOGO	2.3	OSCLI	5.7
Masterfile	5.7	OS_BreakPt	6.11
Maths Movies	6.2	PC	
Memory dumps	5.8	Compatible	2.2
Memory upgrades	2.11	Emulator	5.3, 5.24
Menu Button	2.11	Emulator Upgrades	6.9
Microwave problems	5.8	GW Basic on emulator	6.9
Minerva Systems	1.4	Podule	3.4, 5.2
Minotaur	3.34	Problems with emulator	6.5
Mode 15, 2401 colours	4.8	Workspace	6.8
MODE 7 – Screen-save program	2.44	Pendulum patterns	4.40
Mode changes	6.7	Podules expansion	2.9
Mode Selection Chart	1.42	POINTER 2	2.8
Modes 18-20	1.5	Pointers	2.41, 6.36
Modules, easy loading	6.6	Print buffers, large	6.6
Modules, plugging in and out	5.7	Printer cables	4.7
Monitor		Printer interface connector	1.18
Information	5.5, 6.10	Printer switches	6.2
Multi-sync	1.5, 2.6, 6.6	Program structure, BASIC V	1.28
Problems with	3.7	Prog Ref Manual	2.5, 3.3, 4.4, 5.11, 6.11
Morley RAM disc	6.5	RAM disk/ filing system	5.3, 6.5
Mouse		RAM upgrades	3.7
Art Routines	5.33	RAMBASIC	1.9, 6.7
Operated CAT	4.29	Reading can be fun	2.34
Pointer designer	5.28	Reading the catalogue	5.7
Use in 65Arthur	4.5	Removing the lid	2.9
MS-DOS & DOS+ to ADFS	4.2	RESOURCE – 6502 conversions	3.6
Multi-sync monitors	1.5, 2.6, 6.6		
Multi-sync output, tesing	6.6		

Reversing the CAPS lock.....	6.7	SYS	5.7
RGB connector.....	1.18, 2.6	System Delta Plus	4.10, 4.32
RMtidy	3.9	SystemDevs Module	3.10, 6.41
ROM Podule, Comp'r Concepts ...	3.5, 5.2, 6.3	Taxan 770+ multisync monitor	6.10
ROM-LINK on Archimedes	1.20	Teletext Adaptor	5.2, 6.14
ROMs, BBC, in 65Arthur	3.10, 4.7	Testing multi-sync output	6.6
RS423 links	5.3	Text editor offer	5.3
RS423 problems	1.10, 2.15, 3.30, 6.9	Text files for BASIC	5.6
RS423 transfer prog	5.4	Text processor/editor	3.6
RTC (Real time clock)		Time format	4.5, 4.6, 6.28
*TIME format	4.5	TRACE, problems with	6.11
Naff RTC Month	2.9	TWIN and TWINO	2.9
Setting the clock.....	2.8, 5.7	Undocumented screen modes	3.9
Time formats in BASIC	4.6	User Group	
SCML Teletext Adaptor	6.14	Bristol.....	4.2
Scottish User Group	3.3	Scottish.....	3.3
Screen Banks – Use of	4.38	South Wales	5.3
Screen dumps	1.9, 2.36, 3.32, 4.20, 5.42	User Guide Errata	1.42, 2.49
Centronics GLP	5.9	User Port on the I/O podule	3.12
Colour	6.2	VDU 23,26	6.5
One liner.....	5.7	Video Digitiser.....	2.3
Screenload/save Module	6.9	View	1.5, 3.9, 5.7, 5.8, 6.7
Screensave, fast	5.6	Whales in Zarch	3.11
Screens into different modes	6.7	Wildcards on ADFS	2.10
Seikosha printer	6.9	WIMP Environment	2.16, 3.26, 5.37, 6.30
Serial links	5.3	WIPE warnings (0.2 & 0.3)	1.7
SETTYPE	3.44	Word Perfect – PC Emulator	4.5
Sideways Scrolling with VDU23	3.39	Word Processors	2.8
Slower listings	6.8	Wordprocessors, file transfer	6.8
Sound effect, interesting	4.7	Wordwise Plus	1.2, 2.4, 2.20
Sounds Interesting.....	3.35	Wordwise Plus boot file	3.11
Special effects in VIEW.....	3.9	Zarch	2.3, 3.11
Sprite editor.....	5.6		
Statistical Software	3.6		
Stereo speakers.....	2.7		
String handling, BASIC V	4.19		
Submitting Articles to Archive	5.50		
SWI – Neater than OSCLI	5.7		
SWI Listings	3.47		
Switching off desktop	4.7		

This index is based on material supplied free of charge by Matthew Treagus (aged 15). If you feel inclined to send him something for his efforts, I'm sure he wouldn't object! (30 Hampton Lane, Blackfield, Southampton, SO4 1ZA or come and see him at the Micro User Show.)

Using the Debugger – Part 2

Gerald Fitton

Introduction

In last month's program, "Memory01", we used the debugger command *MemoryB to display a block of memory as ASCII or hex. The program also provided a facility for modifying any byte of memory by entering changes in ASCII or hex. This month we shall concentrate exclusively on the disassembly and modification of machine code (i.e. executable object code) particularly with the debugger command *MemoryI. To do this we first need some machine code. Perhaps you've already got some, but if not then either you can use the source code program "Message01" to generate some simple object code, or, for the more ambitious, you can use "Memory02" to type in, as floating point mnemonics, the code for calculating the square root of 10 to 18 significant figures.

The Program "Message01"

This is a short (40 line) program which can be used either as a stand alone code generator which stores its object code in a directory called &.ObjectCode or as a LIBRARY file containing the self-contained function FN_Message01 for generating relocatable object code. When the object code is CALLED it prints a message on the screen "Watch this space." followed by about 32 full stops.

The *MemoryA command

This debugger command provides a way of typing hex code directly into memory one (4-byte) word at a time. It disassembles and displays only that one word, it has no facility for entering either ASCII or ARM mnemonics. Perhaps it's a bit strong to say that it's a dead loss, but this month's program offers more facilities: so I shall say no more about *MemoryA.

The *MemoryI Command

This debugger command disassembles and displays a block of memory as five fields across the screen. The first of these fields is the the address in memory of the start of a 4-byte word; the second field displays the value of the contents of those 4 bytes as

printable ASCII characters; the third contains the hex code of the word in "reverse order" (i.e. the 4th, high byte, first), by now you will have remembered (!) that words are placed in memory with the low byte at the lower address; and the fourth field contains the assembler mnemonics. The debugger disassembles and displays not only ARM mnemonics but also floating point mnemonics.

Brief Description of Program Memory02

This program represents a considerable extension to, and some modification of the program "Memory01". Many of the modifications to the procedures FNalterASCII and FNchangehex arise because the ASCII and hex fields are interchanged and because the hex field is "reversed" as explained. A new pair of procedures, FNchangemnemonic (line 11600) and FNinsertmnemonic (line 11900), have been added to allow entries to be made into the mnemonic field. Also added is the procedure FNreadmnemonic (line 11500) which uses OS_Byte 135 to read the mnemonic as it is displayed on the screen and PROCassemble (line 25000) which is used to assemble object code from an ARM mnemonic. This last procedure also contains a LOCAL ERROR trap which "vectors" mnemonics not recognised by BASIC to the procedure PROCfloatingpoint (line 20000). This floating point "assembler" is not intended to be comprehensive but it is included to show how a BASIC program containing ARM mnemonics can be extended to include f.p. mnemonics (or any new co-processor mnemonics). With the three mnemonics of PROCfloatingpoint the square root of 10 can be evaluated.

What Can You Do With the Program?

My general philosophy for programming is to write them in three sections. The first (pre-core) section uses lines 100 upwards to declare variables and describe, with REM statements, how they will be used in the program. My core section, from line 1000 onwards I try to keep as short as possible (but not this time!) and use it only to call self-contained procedures which do the work. These procedures

(the third section) start at line 10000 with DEF PROCError (in case I forget END - what if the first procedure was "Delete all my precious files?").

So, it is with a half apology that I am using over 80 lines of core section. My excuse is that I think in this case it helps the reader to see what they can do with the procedures contained within the program. A better programming technique would be to include the routines (e.g. the choice of file to load) as FNs which would be selected from a single keystroke (or mouse selected) menu.

Lines 1050 to 1140 contain a routine so that you can move around the hierarchy of directories looking at the files they contain. You can use "A" to move up the directory tree. The routine between 1150 and 1230 lets you select an object code (or other) file to load into the memory at &10000 and lets you know the size of the file. You must use the routine at lines 1250 to 1310 to set the buffersize at least as large as the file you are going to load. It doesn't matter if it's too big. It must not be too small. The "clever bit" at 1280 to 1300 will guarantee that the buffer starts at &10000, which can be useful when reading addresses in the code.

As an alternative to loading an existing object code file you can use lines 1350 to 1430 to load a self-contained code-generating FN such as "Message01" into the LIBRARY, run it and assemble the code into the buffer. You must be in the directory containing the LIBRARY file for this to work, and the filename must be the same as the name of the function (in this case "Message01") and this name is case dependent. If the assembled code is longer than the buffer, the program will crash at this point.

Line 1460 repeatedly calls FNmodify which is the entry point to all the memory modifying functions (see below). You can CALL the code (lines 1500 to 1570) before saving it. If the program doesn't crash but just produces the wrong effect you can re-run the program (loading nothing into the buffer) and make modifications before calling the code again. Finally you can save the executable object code in the (existing) \$.ObjectCode directory (or not!) by any file name you choose after selecting how much of the buffer you wish to save.

A *DUMP of the resulting object code file isn't essential but you may like it. The routine at line 1800 can be modified to send the disassembled buffer to a printer. If you want to do this then change 1800 to VDU 2 : PROCmemory("I ", buffer%, savefilesize%,0,0):VDU 3.

Those of you with programming skills (all Archive readers?) shouldn't find it too hard to put REPEAT - UNTIL loops around the routines you want to repeat. The most ambitious will, no doubt, make it all run from DeskTop!

FNmodify and its Dependants

Pressing <Tab> changes tab% from 0 to 1, from 1 to 2 and from 2 back to 0. The lines following 11240 select which of the ASCII, hex or mnemonics fields can be modified. Of the FNs called, FNalterASCII and FNchangehex work in a manner similar to those of the program "Memory01" of last month. Save the program, switch off the computer and then LOAD and RUN the program. This guarantees an empty buffer. Just press <return> until PROCmemory is called, displaying a block of memory extending from &10000 to the word after &1003C. Type a few characters into the ASCII field. Note that values in the hex field change "backwards", from the right. After four characters, the cursor appears to move down a line and you will see in the previous line *MemoryI has disassembled the code into the corresponding mnemonics. If you press <Tab> you can enter code in hex. Note that now the ASCII characters are entered "backwards". You can enter all zeros in the hex field to generate a machine code instruction, "ANDEQ R0,R0,R0", that in effect, does nothing.

As the next experiment you can enter a short program in machine code which will print a message on the screen. Clear a block of memory starting at address &10000 by entering zeros in the hex code field. Starting at address &10000 enter "EF020001" which has the mnemonic "SWI XOS_Writes". This is the code for "Write the ASCII text string following this instruction until you get to a zero byte". Starting at address &10004, and in the ASCII field, type in some text such as "Watch this space." etc. Make sure you leave some hex 00s after the end of the text (the "end of string"

marker), then type, in the hex field, at the next full hex word after the zeros, the code "E1A0F00E". This will produce the mnemonic "MOV PC,R14" which is the code to return to BASIC after CALLing a machine code program. Having done all this, if you press <return> you will be asked if you want to CALL the code, press "Y" and your text will be printed to the screen. You are now a genuine "Machine Code" programmer.

Now for something harder. Switch off and start again. After typing in the code "E1A0F00E" move into the mnemonics field where "MOV PC,R14" is found and change it to "MOV R0,R14". If you now <tab> either you will find that the hex code has changed to "E1A0000E" (and the mnemonic has been accepted) or the program will crash. Press <Escape> if it worked (or <reset> followed by OLD if it crashed) and list lines 25000 to 25120. You will find that line 25090 has changed from 50 semicolons to "MOV R0,R14". This is the self-modifying feature of the program executed by line 25030.

If your program crashed then you should check that you have typed in lines 25090 onwards exactly with no spaces after the line numbers (Easier to buy the program disc, I think. Ed.) and exactly 50 semicolons in line 25090. The value &43 in line 25030 is chosen to place the amended mnemonic in exactly the right place. For those having real trouble, either give up and buy the disc from Paul, or change line 25030 to \$(TOP-&40)=LEFT\$ (mnemonic\$,40) and work out what has gone wrong, gradually increasing the &40 until the mnemonic starts immediately after the line number. You can then change the right hand side of the equation of line 25030 back to LEFT\$ (mnemonic\$,50).

There are some anomalies to look out for when entering ARM mnemonics. The SWI instructions must be entered with inverted commas such as SWI "XOS_WriteS" even though the mnemonic display of *MemoryI does not show the inverted commas. This results in code corruption if you <Tab> out of a mnemonic field having a SWI instruction. The program will not recognise BASIC commands which have to be tokenised before presentation to

the BASIC compiler. An example of a mnemonic which fails is MOV R0,#ASC("A"). From within our program ASC(is not tokenised when it is presented to line 25090. It would have been tokenised if the line were entered from the keyboard and the BASIC compiler expects this.

The next piece of machine code will "work" only if you have RMLOADED the f.p.e. (version 2.40 not the earlier one, which won't work). It uses the f.p.e. to find the square root of 10 to 18 significant figures. Start the code at address &10000. In the hex field enter the code "EE08810F", the mnemonic displayed is the f.p. mnemonic "MVFE F0,#10.0" which moves the value 10.0 into the f.p. register F0. In the next line, address &10004, enter "EE489100" which gives the f.p. mnemonic "SQTE F1,F0", for placing the extended double precision square root of the number held in F0 into F1. "E28F0004" in address &10008 places the address &10014 in the ARM register R0, which, together with the next two words, will be used to store the answer. Enter "ECC09100" into address &1000C: this code transfers the calculated square root (held in F1) as Packed Binary Coded Decimal into the address held in R0, i.e. into &10014 and the next two words. Finally "E1A0F00E" in &10010 will return control to BASIC. Press <Return>, CALL the code and save the file as "Root" with a savefilesize of &20. The disassembled code (line 1800) from address &10014 will show the numbers 000003162277660168379332. This should be interpreted as 3.162277660168379332 and is the square root of 10 correct to about 18 sig. figs.

The program "Memory02" includes the procedure PROCfloatingpoint (line 20000) in which lines 20030 to 20050 are designed to "trap" and "assemble" the f.p. codes of the "Root" program to be entered as f.p. mnemonics. Clear the memory from &10000, enter into the mnemonics field "MVFE F0,#10.0" being careful to have 4 spaces between the E and the F. Press the down arrow and the machine code "EE08810F" should appear. Similarly, the mnemonic "SQTE F1,F0" will produce "EE489100" through line 20040 of the program. In the next line type "ADR R0,buffer%&14" which is the mnemonic for

"Place in the register R0 the address of buffer%+&14 (i.e. &10014) relative to the current address". Note that &10014 is the address where the answer will be stored. The disassembled mnemonic is "ADD R0,PC,#4" (see Peter Cockerell's book for details). The final two mnemonics of the program are "STFP F1,[R0],#0" and "MOV PC,R14". The program can now be run and stored to provide the same answer as before.

In Conclusion

Well, hopefully you have learned quite a bit more about the debugger command *MemoryI (and the rather less useful *MemoryA) as well as about self-modifying programs. If you have typed in "Message01" then you can assemble, modify, run and save its code from within "Memory02". I feel sure you can see the potential of this method. For those of you buying the disc from Norwich Computer Services there are also two programs "&.SourceCode.Header01" and "02" for generating simple relocatable module headers. These can be used as stand alone programs which create and RMLOAD a complete module or they can be used as LIBRARY functions with "Memory02". If you have a machine code program without source code, you will be able to use "Memory02" to modify it once you have decided how it works. Finally, the program has the potential for extending PROCfloatingpoint to "assemble" the whole set of f.p. mnemonics.

For my part, I would like some feedback through the magazine. Has anyone found a clever, legal way of using the BASIC assembler to assemble a single line of code without using a self-modifying program? Can the "look-up" table in the debugger which converts code to mnemonics be used in reverse to generate compiled code? This would help with the f.p. mnemonics.

I shall have a short rest and wait for comments before embarking on the use of the debugger's breakpoint and showregs commands. See you in a month or two! **A**

(The problem with these program listings is that to get the self-modifying effects Gerald speaks of, you need to type it in exactly as he did. However, for reasons of space I have had to reduce the listings by cutting out spaces etc. If you want a copy of the programs and begrudge spending £3 for the program disc, send a formatted disc and a return envelope and we will copy these listings for you as supplied by Gerald. Ed.)

```

30000 REM > &.SourceCode.Message01
30010 REM Author : G L Fitton
30020 REM Copyright:ABABCUS TRAINING
30030 REM Version 1.00:31/3/1988
30040 :
30060 REM The source code for a
           simple machine code program
30070 :
30080 DIM code% &200
30090 endofcode%=FN_Message01(code%)
30100 OSCLI("SAVE &.ObjectCode.
           Message01 "+STR$(code%)+
           "+STR$(endofcode%))
30110 *SETTYPE &.ObjectCode.Message01
           &PFC
30120 *STAMP &.ObjectCode.Message01
30130 END
30140 :
30160 DEF FN_Message01(start%)
30180 LOCAL sp,pass%
30190 LOCAL code
30210 :
30220 LOCAL ERROR
30230 ON ERROR OSCLI("FX 3,0"):VDU
26,12:REPORT:PRINT " at line ";ERL
           :STOP
30240 :
30250 REM Use the BASIC stack.
30260 sp = 13 :REM Stack pointer.
30270 :
30280 FOR pass% = 0 TO 3 STEP 3
30290   P%=start%
30300   [OPT pass%
30310   .code
30320   STMFD (sp)!,{R0-R12,R14}
30330   SWI "XOS_NewLine"
30340   SWI "XOS_WriteS"
30350   EQU$ "Watch this space.....
           ..... "
30360   EQU$ 0
30370   ALIGN
30380   SWI "XOS_NewLine"
30390   LDMFD (sp)!,{R0-R12,PC}
30400   ]
30410 NEXT pass%
30420 =P%
```



```

100 REM > &.BasicProgs.Memory02
110 REM Author : G L Fitton
120 REM Copyright: ABACUS TRAINING
130 REM Version 1.03:1/4/1988
140 :
160 ON ERROR PROCerror
170 MODE 3
180 :
200 apos% = 11:REM Cursor position
210 avpos% = 10:REM when altering
    memory in ASCII
220 hpos% = 24-(apos%-11)*2:REM
    Position of cursor
230 hvpos% = avpos%:REM when
    changing memory in hex
240 mpos% = 29:REM Cursor position
250 mvpos% = avpos%:REM when
    changing assembler mnemonics
260 display%= FALSE :REM Redisplay
    on screen when TRUE.
270 tab% = 0:REM ASCII, hex or
    mnemonic field.
280 step% = 1:REM Size of step
    through memory
290 dir$=STRING$(10," ")
300 loadfilename$=STRING$(10," ")
310 savefilename$=STRING$(10," ")
320 load%=FALSE
330 save%=FALSE
340 call%=FALSE
350 buffersize%=&100
360 buffersize$=STR$~(buffersize%)
370 savefilesize%=buffersize%
380 :
1000 REM Core Section of Program
1010 REM Many of these routines
    should be replaced by Procs
    or Fns
1020 :
1040 *EX
1050 REPEAT
1060 PRINT "Please select the
    directory you want, ";
1070 INPUT "or press <Return>
    for current." dir$
1080 IF dir$<>" " THEN
1090 PRINT
1100 OSCLI("DIR "+dir$)
1110 OSCLI("EX")
1120 ENDIF
1130 UNTIL dir$=""
1140 :
1150 INPUT "Name of file to load,
    press <Return> for none. "
    loadfilename$
1160 IF loadfilename$="" THEN
1170 load%=FALSE
1180 ELSE load%=TRUE
1190 channel%=OPENIN(loadfilename$)
1200 loadfilesize%=EXT#channel%
1210 PRINT "loadfilesize% = &";
    STR$~(loadfilesize%)
1220 CLOSE#channel%
1230 ENDIF
1240 :
1250 PRINT "The buffer size is &";
    buffersize$;
1260 INPUT ". Press <Return> or
    enter new size &" buffersize$
1270 IF buffersize$<>" " buffersize%
    =EVAL("&"+buffersize$)
1280 DIM dummy1% 0
1290 DIM dummy2% (&FFD8-dummy1%)
    :REM Makes the buffer start at
    &10000.
1300 DIM buffer% buffersize% :REM
    Reserve space in memory.
1310 start% = buffer%:REM Start of
    display.
1320 :
1330 IF load%=TRUE THEN OSCLI
    ("*LOAD "+loadfilename$+" "+
    STR$~(buffer%))
1340 :
1350 PRINT "Name the assembler code
    file to load into LIBRARY (and
    buffer)."
1360 INPUT "Or press <Return> for
    none." libraryfilename$
1370 IF libraryfilename$<>" " THEN
1380 LIBRARY libraryfilename$
1390 filelength%=EVAL("FN "+
    libraryfilename$+"(buffer%
    )")-buffer%
1400 PRINT "filelength% = &";
    STR$~(filelength%)
1410 PRINT "Press any key to
    continue."
1420 wait$=GET$
1430 ENDIF
1440 :
1450 REPEAT
1460 quit%=FNmodify :REM Modify
    bytes in memory.
1470 UNTIL quit%=TRUE
1480 CLS
1490 :
1500 PRINT "Do you want to CALL the
    code before it is saved? (Y/N) "
1510 yesno$=GET$
1520 IF INSTR("Yy",yesno$) THEN call%
    =TRUE ELSE call%=FALSE

```

Using the Debugger

```

1530 IF call%=TRUE THEN
1540   CALL buffer%
1550   PRINT "Press any key to
           continue.  "
1560   wait$=GET$
1570 ENDIF
1580 :
1590 REM Save modified file to disc
1600 *EX &.ObjectCode
1610 INPUT "Name of file to save,
           press <Return> for none. "
           savefilename$
1620 IF savefilename$="" THEN
           save%=FALSE ELSE save%=TRUE
1630 IF save%=TRUE THEN
1640   PRINT "The savefilesize% is
           &;STR$(buffersize%);
1650   INPUT ". Press <Return> or
           enter new size &"
           savefilesize$
1660   IF savefilesize$<>" THEN
           savefilesize%=EVAL("&"
           +savefilesize$) ELSE
           savefilesize%=buffersize%
1670   CLS
1680   OSCLI("SAVE &.ObjectCode."
           +savefilename$+" "+STR$(
           (buffer%)+ " "+STR$(
           (savefilesize$)+ " 0 0")
1690   OSCLI("SETTYPE &.ObjectCode."
           +savefilename$+" &FFC")
1700   OSCLI("STAMP &.ObjectCode."
           +savefilename$)
1710   PRINT "The file called
           &.ObjectCode.";savefilename$;
           " now contains:-"
1720   OSCLI("DUMP &.ObjectCode."
           +savefilename$)
1730   PRINT
1740   ELSE
1750   CLS
1760 ENDIF
1770 :
1780 PRINT "Press any key to
           continue."
1790 wait$=GET$
1800 PROCmemory("I ",buffer%,
           savefilesize%,0,0)
1810 PRINT "Press any key to
           continue."
1820 wait$=GET$
1830 CLS
1840 END
1850 :
10000 DEFPROCerror
10010 CLOSE#0
10020 *FX 4,0
10030 CLS
10040 REPORT
10050 PRINT " at line ";ERL
10060 END
10070 :
10080 ENDPROC
10090 :
10200 DEF PROCmemory(type$,pstart%
           ,length%,x%,y%)
10210 REM Displays block of memory.
10220 LOCAL command$
10230 CLS
10240 PRINT TAB(x%,y%);"Address
           ASCII Hex Code Assembler
           Mnemonics"
10250 command$="Memory"+type$+" "+
           STR$(pstart$)+" "+STR$(
           (length%))
10260 OSCLI(command$)
10280 ENDPROC
10290 :
10300 DEF FNalterASCII(RETURN apos%,
           RETURN avpos%)
10310 REM Lets user change the memory
10320 LOCAL key%,key$
10330 :
10340 REPEAT
10350   PRINT TAB(apos%,avpos%);
10370   REM Cursor keys take CHR$
           values
10380   *FX 4,1
10390   key%=GET
10400   *FX 4,0
10410   key$=CHR$(key%)
10420   IF INKEY(-1)=TRUE THEN
           step%=8 ELSE step%=1
10430   IF key%>31AND key%<127 THEN
           PROCinsertASCII(key%,apos%
           ,avpos%)
10440   IF key%=136 THEN apos%=apos%-1
10450   IF key%=137 THEN apos%=apos%+1
10460   IF key%=138 THEN start%=
           start% + 4*step%:
           display%=TRUE
10470   IF key%=139 THEN start%=
           start% - 4*step% :
           display%=TRUE
10480   IF key%=9 THEN tab%=(tab%+1)
           MOD3:display%=TRUE
10490   IF key%= 13THEN quit%=TRUE
           :display%=TRUE
10510   :
10520   IF apos%=10 THEN apos%=14:
           start%=start% - 4:
           display%=TRUE

```



```

10530 IF apos%=15 THEN apos%=11:
      start%=start% + 4:
      display%=TRUE
10540 :
10550 UNTIL display%=TRUE
10560 display%=FALSE
10580 =quit%
10590 :
10600 DEF PROCinsertASCII(key%,
      RETURN apos%,RETURN avpos%)
10610 REM Modifies the memory and
      prints to the screen.
10620 LOCAL key$,hex$
10630 key%=CHR$(key%)
10640 hex%=RIGHT$("00"+STR$(key%),2)
10650 ?(start%+&20+apos%-11)=key%
10660 PRINT key$;TAB(26-(apos%-10)
      *2,avpos%);hex$;
10670 apos%=apos%+1
10680 ENDPROC
10690 :
10700 DEF FNchangehex(RETURN hpos%,
      RETURN hvpos%)
10710 REM Lets user change memory.
10720 LOCAL key%,key$
10740 REPEAT
10750 PRINT TAB(hpos%,hvpos%);
10770 REM Cursor keys take CHR$
      values
10780 *FX 4,1
10790 key%=GET
10800 *FX 4,0
10810 key%=CHR$(key%)
10820 IF INKEY(-1) THEN step%=8
      ELSE step%=1
10830 IF INSTR("0123456789ABCDEF",
      key%) THEN PROCinsertthex
      (key$,hpos%,hvpos%)
10840 IF key%=136 THEN hpos%=hpos%-2
10850 IF key%=137 THEN hpos%=hpos%+2
10860 IF key%=138 THEN start%=start%
      +4*step%;display%=TRUE
10870 IF key%=139 THEN start%=start%
      -4*step%;display%=TRUE
10880 IF key%=9 THEN tab%=(tab%+1)
      MOD3:display%=TRUE
10890 IF key%=13 THEN quit%=TRUE
      :display%=TRUE
10910 :
10920 IF hpos%=16 THEN hpos%=24:
      start%=start% - 4:
      display%=TRUE
10930 IF hpos%=26 THEN hpos%=18:
      start%=start% + 4:
      display%=TRUE
10940 :
10950 UNTIL display%=TRUE
10960 display%=FALSE
10980 =quit%
10990 :
11000 DEF PROCinsertthex(key1$,RETURN
      hpos%,RETURN hvpos%)
11010 REM Modifies the memory and
      prints to the screen.
11020 LOCAL key2$,hex$
11030 PRINT key1$;
11040 REPEAT
11050 key2$=GET$
11060 IF INSTR("0123456789ABCDEF"
      ,key2$)=0 THEN VDU 7
11070 UNTIL INSTR("0123456789ABCDEF"
      ,key2$)
11080 PRINT key2$
11090 hex%=key1$+key2$
11100 hex%=EVAL("&"+hex$)
11110 IF hex%>31 AND hex%<127 THEN
      char%=CHR$(hex%) ELSE
      char%="."
11120 PRINT TAB(14-(hpos%-18)DIV2,
      avpos%);char$;
11140 ?(start%+&20-(hpos%-24)DIV2)
      =hex%
11160 hpos%=hpos%+2
11180 ENDPROC
11190 :
11200 DEF FNmodify
11220 PROCmemory("I ",start%,&40,0,0)
11230 PROCchelp
11240 CASE tab% OF
11250 WHEN 0
11260 apos%=11-(hpos%-24)DIV2
11270 quit%=FNalterASCII(apos%
      ,avpos%)
11280 hpos% = 24-(apos%-11)*2
11290 WHEN 1
11300 hpos% = 24-(apos%-11)*2
11310 quit%=FNchangehex(hpos%
      ,hvpos%)
11320 apos%=11-(hpos%-24)DIV2
11330 WHEN 2
11340 mpos%=29
11350 mnemonic$=FNreadmnemonic
      (mpos%,mvpos%)
11360 quit%=FNchangemnemonic
      (mpos%,mvpos%)
11370 ENDCASE
11380 =quit%
11390 :
11400 DEF PROCchelp
11410 PRINT TAB(0,21);
11420 PRINT "Press cursors to move
      through memory."

```

Using the Debugger

```

11430 PRINT "Press <Shift> with
        cursors to move faster."
11440 PRINT "Press <Tab> to toggle
        between ASCII, hex and
        assembler mnemonics."
11450 PRINT "Enter changes at cursor
        in ASCII or hex. ";
11460 PRINT "Press <Return> to
        finish.";
11480 ENDPROC
11490 :
11500 DEF FNreadmnemonic
        (mpos%,mvpos%)
11510 LOCAL pos%,chr%,mnemonic$
11520 PRINT TAB(mpos%,mvpos%);
11530 FOR pos%=mpos% TO mpos%+49
11540   SYS "OS_Byte",135 TO ,chr%
11550   mnemonic$=mnemonic$+CHR$
        (chr%)
11560   VDU 9
11570 NEXT pos%
11580 =mnemonic$
11590 :
11600 DEF FNchangemnemonic(RETURN
        mpos%,RETURN mvpos%)
11610 REM Lets user change memory.
11620 LOCAL key%,key$
11630 :
11640 REPEAT
11650   PRINT TAB(mpos%,mvpos%);
11670   REM Cursor keys take CHR$
        values
11680   *FX 4,1
11690   key%=GET
11700   *FX 4,0
11710   key$=CHR$(key%)
11720   IF INKEY(-1) THEN step%=8
        ELSE step%=1
11730   IF key%>31 AND key%<127 THEN
        PROCinsertmnemonic(key$
        ,mpos%,mvpos%)
11740   IF key%=136 THEN mpos%=mpos%-1
11750   IF key%=137 THEN mpos%=mpos%+1
11760   IF key%=138 THEN PROCassemble
        :start%=start%+4*step%
11770   IF key%=139 THEN PROCassemble
        :start%=start%-4*step%
11780   IF key%=9 THEN PROCassemble
        :tab%=(tab%+1)MOD3
11790   IF key%=13 THEN PROCassemble
        :quit%=TRUE
11810   :
11820   IF mpos%=28 THEN mpos%=78:
        PROCassemble:start%=start%-4
11830   IF mpos%=79 THEN mpos%=29:
        PROCassemble:start%=start%+4
11850 UNTIL display%=TRUE
11860 display%=FALSE
11880 =quit%
11890 :
11900 DEF PROCinsertmnemonic(key$
        ,RETURN mpos%,mvpos%)
11910 REM Modifies the memory and
        prints to the screen.
11930 MID$(mnemonic$, (mpos%-28),1)
        =key$
11950 PRINT TAB(29,mvpos%);mnemonic$;
11960 mpos%=mpos%+1
11970 PRINT TAB(mpos%,mvpos%);
11980 ENDPROC
11990 :
20000 DEF PROCfloatingpoint
20010 REM A skeleton FP assembler
20020 CASE LEFT$(mnemonic$,19) OF
20030   WHEN "MVFE F0,#10.0 " :
        ! (start%+&20)=&EE08810F
20040   WHEN "SQTE F1,F0 " :
        ! (start%+&20)=&EE489100
20050   WHEN "STFP F1,[R0],#0 " :
        ! (start%+&20)=&ECC09100
20060 ENDCASE
20070 display%=TRUE
20080 ENDPROC
20090 :
25000 DEF PROCassemble
25010 LOCAL ERROR
25020 ON ERROR LOCAL
        PROCfloatingpoint:ENDPROC
25030 $(TOP-&43)=LEFT$(mnemonic$,50)
25040 display%=TRUE
25060 P%=start%+&20
25070 [OPT 0
25080 .store
25090 ;;;;;;;;;;;;;;
        ;;;;;;;;;;;;;;
25100 ]
25110 ENDPROC A

```


Beginner's Guide to Fonts

Keith Milner

Introduction to Fonts

Bundled with the Archimedes is a system which allows a variety of typefaces, or "fonts" to be loaded in and painted on the screen. The Acorn font system allows any size (within reason) of character. What is more, the characters are proportionally spaced (this means that for instance the letter 'i' takes up less space than the letter 'm'), and may be justified.

The Acorn fonts are also anti-aliased. This is where the edges of a shape are 'fuzzed' which creates the impression of a higher screen resolution and makes the characters easier to read, especially when using small point sizes. It has the disadvantage that larger point sizes look a bit ugly. (Point size is a typo-graphical term used to describe the size of the characters.)

There are several parts to the Acorn font system: there is a font designer to enable you to create the fonts; there is the anti-aliasing program to 'fuzz' the edges; there is the font manager to handle access to the fonts; and finally the font painter draws the font on the screen. The first two are supplied on the Welcome disc, the second two are built into the operating system (on 1.2 upwards). This article covers the use of fonts in your own programs.

Preparation

Before leaping into using fonts, you will need to prepare a disc containing the font files. (Or purchase the program disc!) If you are using a single disc system, don't try to *COPY the FONTS folder because you will find it needs about 16 swaps of source and destination disc. The easiest way is to backup the Welcome disc onto a freshly formatted blank disc. This is done by exiting from the desktop and typing: *BACKUP 00Q. Then go back into the desktop

with *DES. or somesuch, and delete all but the FONTS folder. This is done by marking each of the other folders by clicking on each one with the select button, then pree <menu> and select Delete and then confirm each deletion in turn.

The next job is to create a new directory for the programs, either by pressing <menu> and selecting "New dir." or by exiting from the desktop and typing *CDIR &.FontProgs.

You are now left with a working disc containing the font files within the directory "fonts" and a directory "FontProgs" to put the programs into.

Getting Started

Before we use the fonts, we need to know what styles are available to us. One of the commands built into the font manager is FONTCAT. This gives a description of all the font files within any directory. To use it just type:

```
*FONTCAT &.fonts
```

On my version of the Welcome disc, this gives the following output:

```
Corpus.Medium  
Trinity.Medium
```

To save the bother of specifying a directory every time that FONTCAT (or indeed any other font command) is used, a prefix may be set. This is done by typing: *SET Font\$prefix &.FONTS

Once this is set, you can just type *FONTCAT to obtain a list of available fonts.

Managing Fonts

This is done automatically by the font manager module. When you ask the manager to "find" a font, it will search the disc for the specified font files (in the directory specified by Font\$prefix). If it finds the file, it loads information about the font and where it may be found on the disc into an area of private memory called the "cache".

When you ask the font system to display some characters in that font, the manager will load a block containing the required characters into the cache. Only the parts of the character set which are needed are loaded in. This maximises the memory available to other fonts. If you try to load in more fonts than the cache has room for, the manager will erase a previously used font. Next time you try to access that font, the manager will have to load it in again.

You may set the cache size by typing:

```
*CONFIGURE FONTSIZE n
```

where n is the number of pages you require. On the 300 series each page is 8K. A good value to set it to for this tutorial is 10 (this reserves 80K). 400 series owners may set this number to 10 as well. This actually reserves 320K on a 4M machine, but with that much memory to play with, what's 320K?

After setting this value, the machine should be reset with <ctrl-break> before further use.

Font Demonstration Program

By now, you will be itching to get something on the screen. Type in listing 1 (Font_demo) and run it. This will paint a message on the screen in one of the fonts. Notice how slowly this happens. This is because the font is being drawn in from the disc. If you run the program a second time, the message will be painted immediately – the font has been stored in the cache and is thus available for instant access.

You may discover which fonts are known to the font manager by typing: *FONTLIST

This displays a list of all the recognized fonts with information about their size and usage.

The program listing has been made deliberately long in order to show the function of its various parts. In practice, this program could be considerably shortened. The functions of the various parts of the program are summarized:

line 90	sets up the font prefix as described
line 120	the name of the font to be used
line 150	sets the x and y point sizes of the font to be used
line 180	sets the resolution of the screen
line 210	asks the manager to load the font information using the data set up in the previous lines
line 230	selects 640 x 256 16 colour mode
line 260	onwards, sets up the anti-aliasing transfer levels.
line 410	onwards, sets up the anti-aliasing palette
line 590	gives the string to be painted
line 620	paints the font onto the screen

Now we need to look in more detail at the main features of this program.

Point Sizes

The size of the character is determined by its point size. One point is 1/72 of an inch. The font system allows you to specify point sizes to within 1/16 of a point (i.e. 1/1152 of an inch). It also allows you to specify different X and Y point sizes to give, for instance, tall thin characters. Try setting x_points to 20 in program 1, Font_demo.

This seems to give an almost infinite range of sizes from one font style. There is however one catch. The fonts are internally stored in several standard point sizes. If you display a font in one of these sizes, it will be loaded in as stored. If, however, you choose a point size for which there is no data stored, the font manager will make sensible "guesses" about what the character should look like. This does result in a slight reduction in quality but in practice this is not usually obvious. Larger point sizes, however, do look ragged and should be avoided.

Screen Resolution

This specifies the resolution (in pixels per inch) of the screen. In normal use the default value of

X=90, Y=45 will be used. If however a higher resolution monitor (a multi-sync, say) is used, changing this value will maintain the correct aspect ratio for the characters. For instance, in mode 20 (640x512) you could use X=90, Y=90. Apart from this, there is little practical purpose in altering these numbers. When you load a font, the manager will look for a file containing data in the resolution you specify. If it can find none, it will take the data for the nearest equivalent resolution and use that. Currently only data for X=90, Y=45 is supplied on the Welcome disc, so changing these numbers achieves little.

Find Font

This is a SYS call to the font manager. I have chosen to pass the data as variables, as it is much neater. If desired, however, the data could be passed directly, as follows:

```
180 SYS "Font_FindFont",,"Trinity.Medium"
    ,40*16,40*16,0,0 TO Trinity%
```

The font manager allocates a number or "handle" to the font, so that it may easily be referred to at a later date. This is passed back from the routine into a user-specified variable. I have called this Trinity% for obvious reasons.

Transfer Level & Anti-aliasing Palette

This requires an understanding of how the anti-aliasing is achieved. Running the second programs, Anti_alias, will give a graphic demonstration of this: a message in 40 point Trinity font is displayed with the first letter "A" magnified. The characters are stored with 16 levels of intensity, called "transfer levels" varying from the background colour, in this case cyan, to the foreground colour, which I have defined to be black. When the font is painted onto the screen, the computer assumes that the palette has been set up to give this graduation, assigning the background to colour 0, the foreground to colour 15, and all the transfer levels in between to the intermediate colours.

In Anti-alias, the characters have been painted before the palette was correctly set up. With the default palette for this mode, the character appears surrounded by multi-coloured and flashing blocks. Pressing the space bar will toggle the anti-aliasing palette on and off so that you can see the relationship between the intensity levels and the colours.

There are, however, cases where you don't wish all 16 colours to be used. In this case you may override the default values and specify that only 8 or 6 or any other number of colours will be used for anti-aliasing (the minimum you can use is 2 colours, but the resulting picture is not too good!!). This is done with a SYS call which specifies how many colour registers to use and manually sets the transfer level for each one. A more limited version of this call is available at present as a VDU sequence, but Acorn have made it clear that future versions may not include any of the font VDU calls. For this reason, I am sticking firmly to using SYS calls.

You may set the transfer levels to any values you wish but it is normal to set them so that the colours cover all 16 transfer levels evenly. Running the third program, Trans_levs, will draw a series of bars showing such values of transfer function for 16 (default), 8 and 4 colours. The SYS call is made with a table of values stating how many colours are to be used and the transfer level for each colour. When specifying the number of colours, you actually specify the "colour offset". What this amounts to is that the font system always assumes you will need one background and one foreground colour. You specify how many you will need above this. e.g. to specify 8 registers, you set the colour offset to 6. The best method to generate such a table is via a DIM statement. We can then insert the name of the block of reserved memory directly into the SYS call.

We may now define the anti-aliasing palette. To do this we must decide what colour the foreground is to be, what colour the background will be and which registers we wish to use. If you are using, for instance, 4 colour registers for anti-aliasing, you may specify to use colour 0 as the background colour and colours 8 to 10 as the three foreground colours. The format of the SYS call is:

```
SYS"Font_SetPalette",,bcol_reg,  
fcol_reg,fcol_offset,bcol,fcol
```

where bcol_reg is the background colour register

fcol_reg is the 1st foreground register

fcol_offset is the number of additional colour registers to use

bcol is the physical background colour

fcol is the physical foreground colour

The **physical** colours are passed as a 32 bit number which contains information about the red, green, and blue components of the actual colour you want. This is stored as a hex number: &BBGGRR00

where BB, GG, & RR are hex numbers representing the blue, green, and red components of the desired colour. As an example, if the background colour is to be cyan (RED=0, GREEN=&FF, BLUE=&FF) then bcol would be &FFFF0000; fcol=&00000000 means the foreground will be black. If we then set bcol_reg=0, fcol_reg=1, and fcol_offset=2 then the palette will be set up with 4 colours ranging from colour 0 being cyan to colour 3 being black.

Font Paint

Finally, after all of this we can display the message on the screen. This is done using the SYS call at line 620 of the original Font_demo program. This takes the following form:

```
SYS"Font_Paint",,message$,type  
x_pos,y_pos
```

where message\$ is the string to be painted.

type specifies the type of painting to be done.

x_pos & y_pos are the x/y screen positions to paint the message.

The plot type is a byte containing several flags which specify how the font should be painted on the screen. The bits defined are as follows:

Bit	Action
0	Justify text
1	Rub-out box required
2	Use absolute co-ordinates
4	Co-ordinates are OS co-ordinates

The absolute co-ordinate bit specifies whether the font will be painted at the absolute position specified by x_pos and y_pos, or whether it will be painted at x_pos, y_pos relative to the current graphics cursor position. This option, however, appears not to function. Absolute coordinates are used whether this bit is set or reset.

If OS co-ordinates are not specified, then x_pos & y_pos must be given in units of 1/72000 of an inch, otherwise the normal PLOT co-ordinate system must be used (1280x1024).

The final program, Paint_type, gives an example of using rub-out and justify. If rub-out is specified, before the string is painted, the font painter will erase to background colour an area of screen. This area is specified by performing a MOVE to the two opposite corners of the rectangular area you wish to be cleared. If justify is required, you must specify the right margin to justify to. This is done by performing a single MOVE to the right margin position. NOTE that this MOVE should have the same Y value as the string you wish to plot. If this is not so, the text will "creep" vertically. Try, for instance, changing listing 4 line 410 from MOVE 900,260 to MOVE900,100 and see what a mess it makes!

If you wish to use both rub-out and justification, you must perform the rub-out MOVES first, followed by the justify MOVE and then paint the string.

It is worth mentioning that none of these calls take the length of the string into account. If the string is too long, it will either disappear off the edge of the screen, or if justifying, overprint itself! There are ways to deal with this, automatically splitting strings into paintable chunks but I will not cover this for the moment as it is quite complex.

How do you know what size of rub-out box you should use? Well, you can either guess or you can use SYS"Font_StringBBox" which returns the X & Y size of a rectangle which exactly surrounds the string in 1/72000 of an inch. You can then convert these to OS coordinates for plotting purposes:

```
SYS"Font_StringBBox",,S$ TO
      ,,,xp%,yp%
SYS"Font_ConverttoOS",,xp%,yp%
      TO ,x%,y%
```

x%, and y% will be returned in standard coordinate units. These can then be used for the rub-out box. It is probable that a small margin will be needed around the box, so bear this in mind and add on a few pixels when performing the MOVES.

Using more than one font

So far I have just explained how to use one font at a time. You may, however, wish to use more than one style or size of font at a time. This is where the "font handle" comes in. This is assigned by the font manager when SYS"Font_FindFont" is called and uniquely identifies the different fonts. To specify which font to use, there is a SYS call:

```
SYS"Font_SetFont",font-handle
```

where font-handle is the number of the font as allocated by the manager. Examining the third program, Trans_levs, will show its use.

Now it's up to you

You should now be able to incorporate fonts in your own programs. Start off with some simple programs of your own and don't be afraid to experiment – it's the best way to learn.

```
10 REM >$.FontProgs.Font_demo
20 REM Anti-aliased font demo
30 REM by Keith Milner
40 REM
50 REM Reserve a block of memory
   for threshold definitions
60 DIM threshold%,17
70
80 REM Set font prefix so the font
   manager knows where fonts
   are on disk
90 *SET Font$prefix &.Fonts
100
110 REM Choose font to be displayed
120 font$="Trinity.Medium"
130
140 REM Choose font point size
150 x_points=40:y_points=40
160
170 REM Choose screen resolution
   (0 gives default x=90,y=45)
180 x_res=0:y_res=0
190
200 REM Ask manager to find the font
210 SYS"Font_FindFont",,font$
   ,x_points*16,y_points*16,x_res
   ,y_res TO trinity%
220
230 MODE12 : REM 16 colour mode
240
250 REM Set up threshold table for
   anti-aliasing
260 n_cols%=8 : REM number of colours
   to be used in anti-aliasing
270 ?threshold%=n_cols%-2
   :REM colour offset (foregnd &
   backgnd assumed)
280 threshold%?1=2:REM 1st threshold
   value
290 threshold%?2=4 :REM 2nd
```

Using Anti-Aliased Fonts

```

300 threshold%?3=6      :REM 3rd
                        threshold value
310 threshold%?4=8      :REM 4th
320 threshold%?5=10     :REM 5th
330 threshold%?6=12     :REM 6th
340 threshold%?7=14     :REM 7th
350 ?threshold%=&FF     :REM End of
                        table marker
360
370 REM Pass threshold table to font
                        system
380 SYS"Font_SetThresholds",,
                        threshold%
390
400 REM Set up colours and registers
                        to use
410 bcol_reg%=0 :REM background
                        colour register
420 bcol_R%=0 :REM background RED
                        component
430 bcol_G%=&FF :REM background
                        GREEN component
440 bcol_B%=&FF :REM background
                        BLUE component
450
460 fcol_reg%=1 :REM 1st foreground
                        colour register to use
470 fcol_offset%=n_cols%-2 :REM
                        foreground colour offset
480 fcol_R%=0 :REM foreground RED
                        component
490 fcol_G%=0 :REM foreground
                        GREEN component
500 fcol_B%=0 :REM foreground BLUE
                        component
510
520 REM convert RGB levels to a 32
                        bit no. &BBGRR00
530 bcol%=(bcol_R% << &08)+(bcol_G%
                        << &10)+(bcol_B% << &18)
540 fcol%=(fcol_R% << &08)+(fcol_G%
                        << &10)+(fcol_B% << &18)
550
560 REM Pass data to font system to
                        set up palette
570 SYS"Font_SetPalette",,
                        bcol_reg%,fcol_reg%,fcol_offset%
                        ,bcol%,fcol%
580
590 message$="An anti-aliased font"
600
610 REM Paint message on screen
620 SYS"Font_Paint",,message$,&10,0
                        ,700
630 END

10 REM >$.FontProgs.Anti_alias
20 REM Anti-aliasing demo
30 REM by Keith Milner
40 REM
50 DIM threshold% 17
60 *SET Font$prefix &.Fonts
70 SYS"Font_FindFont",,"Trinity.
                        Medium",40*16,40*16,0,0
                        TO trinity%
80 SYS"Font_FindFont",,"Trinity.
                        Medium",300*16,300*16,0,0 TO
                        close_up%
90
100 MODE12 : REM 16 colour mode
110
120 FORT%=1TO15:COLOURT%:PRINT
                        "Colour ",T%:NEXT
130 ?threshold%=14 :REM 16 colours
                        for anti-aliasing
140 FOR T%=1TO15
150 threshold%?T%=T% :REM colours
                        1-15 for thresholds 1-15
160 NEXT T%
170 threshold%?16=&FF :REM end of
                        block
180 SYS"Font_SetThresholds"
                        ,,threshold%
190
200 SYS"Font_SetFont",trinity%
210 SYS"Font_Paint",,"Anti-aliasing"
                        ,&10,500,100
220 SYS"Font_SetFont",close_up%
230 SYS"Font_Paint",,"A",&10,350,400
240 GCOLOR,7
250 CIRCLE 530,130,50
260 CIRCLE 600,600,350
270 LINE600,250,530,180
280 REPEAT
290 REPEATG=GET:UNTILG=13 OR G=32
300 IFG=13 THEN END
310 SYS"Font_SetPalette",,
                        0,1,14,&FFFF0000,&00000000
                        :REM set A-A palette
320 REPEAT G=GET:UNTILG=13 OR G=32
330 IFG=13 THEN END
340 VDU20
350 UNTILFALSE

10 REM >$.FontProgs.Trans_levs
20 REM Threshold demo
30 REM by Keith Milner
40 REM

```



```

50 DIM threshold% 17
60 *SET Font$prefix &.Fonts
70 SYS"Font_FindFont",,"Trinity.
    Medium",20*16,20*16,0,0
    TO small%
80 SYS"Font_FindFont",,"Trinity.
    Medium",40*16,40*16,0,0
    TO large%
90 MODE12 : REM 16 colour mode
100 OFF
110 ?threshold%=14 :REM 16 colours
    for anti-aliasing
120 FORT%=1TO15
130 threshold%?T%=T% :REM colours
    1-15 for thresholds 1-15
140 NEXTT%
150 threshold%?16=&FF:REM block end
160 SYS"Font_SetThresholds"
    ,,,threshold%
170 SYS"Font_SetPalette",,,
    0,1,14,&FFF0000,&00000000
    :REM set A-A palette
180 SYS"Font_SetFont",large%
190 ul$=CHR$(25)+CHR$(250)+CHR$(10)
200 SYS"Font_Paint",,,ul$+"Typical
    Transfer Levels",&10,150,200
210 SYS"Font_SetFont",small%
220 SYS"Font_Paint",,"Anti-aliasing
    Threshold levels",&10,450,920
230 SYS"Font_Paint",,"No. of",&10,0
    ,930
240 SYS"Font_Paint",,"Colours",&10,0
    ,880
250 VDU5
260 FOR T%=0TO16
270 X%=250+T%*60
280 LINE X%,900,X%,300
290 IF T%<>16 THEN MOVEX%+20,900
    :PRINT,T%;
300 NEXT
310 LINE0,870,1210,870
320 VDU4
330 SYS"Font_SetFont",large%
340 SYS"Font_Paint",,"16",&10,40,750
350 VDU5
360 FORT%=0TO15:GCOL1,T%:X%=250+T%*60
370 RECTANGLEFILLX%,800,60,30
380 GCOL0,15:MOVEX%+20,790
    :PRINT,T%;
390 NEXT
400 SYS"Font_Paint",," 8",&10,40,550
410 FORT%=0TO15STEP2:READ C%:GCOL1
    ,C%:X%=250+T%*60
420 RECTANGLEFILLX%,600,120,30
430 GCOL0,15:MOVEX%+20,590
    :PRINT,C%;
440 NEXT
450 DATA 0,2,4,6,8,10,12,14
460 SYS"Font_Paint",," 4",&10,40,350
470 FORT%=0TO15STEP4:READ C%:GCOL1
    ,C%:X%=250+T%*60
480 RECTANGLEFILLX%,400,240,30
490 GCOL0,15:MOVEX%+20,390
    :PRINT,C%;
500 NEXT:VDU4
510 DATA 0,5,10,15
520 SYS"Font_SetFont",small%
530 Y%=120
540 REPEAT READ S$
550 SYS"Font_Paint",,,S$,&10,50,Y%
560 Y%-=40
570 UNTILY%=0
580 DATA Notice how the transfer
    levels are chosen to cover the"
590 DATA"whole intensity range in
    even steps. "
600 DATA These are suggested
    values only."
610 G=GET:MODE12:END

10 REM >$.FontProgs.Paint_type
20 REM Rub-out box & justification
    demo
30 REM by Keith Milner
40 REM
50 DIM threshold% 17
60 *SET Font$prefix &.Fonts
70 SYS"Font_FindFont",,"Trinity.
    Medium",30*16,30*16,0,0
    TO trinity%
80
90 MODE12 : REM 16 colour mode
100 ?threshold%=6
110 threshold%?1=2
120 threshold%?2=4
130 threshold%?3=6
140 threshold%?4=8
150 threshold%?5=10
160 threshold%?6=12
170 threshold%?7=14
180 ?threshold%=&FF
190
200 SYS"Font_SetThresholds",,,threshold%
210 SYS"Font_SetPalette",,,0,8,6
    ,&FFF0000,&00000000

```

```

210 ul$=CHR$(25)+CHR$(250)+CHR$(4)
220 SYS"Font_Paint",,ul$+"RUB-OUT
    BOXES:",&10,0,950
230 FORT%=1TO100
240 GCOL0,RND(15)
250 CIRCLEFILLRND(1280),RND(800)
    ,RND(200)
260 NEXT
270 SYS"Font_Paint",,"This is text
    without rub-out!!!",&10000,0
    ,700
280 S$="This is text with a rub-out
    box."
290 MOVE0,390:MOVE940,460
300 SYS"Font_Paint",,S$,&10010,0,400
310 MOVE0,190:MOVE900,260
311 SYS"Font_Paint",,"Press -SPACE-
    to continue...",&10010,0,200
312 G=GET
320 CLS
330 SYS"Font_Paint",,ul$
    +"JUSTIFICATION:",&10,0,950
340 SYS"Font_Paint",,"This is
    unjustified text. It",&10000,0
    ,800
350 SYS"Font_Paint",,"has a ragged
    right edge.",&10000,0,730
360 SYS"Font_Paint",,"This makes it
    look messy!",&10000,0,660
370 MOVE900,400
380 SYS"Font_Paint",,"This is
    justified text. It",&10001,0,400
390 MOVE900,330
400 SYS"Font_Paint",,"has its right
    edge aligned",&10001,0,330
410 MOVE900,260
420 SYS"Font_Paint",,"and this makes
    it more tidy.",&10001,0,260
430 G=GET:CLS A

```

Art Packages Review

Malcolm Banthorpe

ArtWorker

Artworker is a graphics program which gives the user the facilities of an air- or spray-brush. It is a deliberately simple art program without the more sophisticated facilities of a program like Artisan such as cut and paste and even lacks such primitives as lines, circles, rectangles etc. It consequently offers little to enhance the limited abilities of the non-artist and relies heavily on human skill to create visually satisfying pictures. The effects produced, though generally referred to as "air-brush" in other graphics programs, are actually more of a pointillist nature and as such can be very effective.

The program is mouse-controlled and operates in graphics mode 15 with 256 colours colours being simultaneously available. Colours are chosen by moving the mouse pointer to the upper edge of the screen. A colour menu showing the 64 basic colours, each in 4 different tints is then displayed at the top of the screen and any one may be selected by moving the mouse to it and pressing <select>. Once the colour menu

has been displayed it is possible to move the mouse pointer elsewhere on the screen and to pick up a colour which already exists in the picture, making it a simple matter to match colours. In addition, any pair of the 64 basic colours may be chosen to be displayed as an alternating pattern of pixels. According to the manual this gives a total of 4288 colours but it doesn't explain how this figure was arrived at. In practice a significant proportion of colour combinations will give a very similar appearance to one of the basic colours.

Moving the mouse pointer to the bottom of the screen reveals a second menu which allows the spray size to be varied. A third menu, accessible from the right-hand side of the screen reveals more spray options and the possibility of saving and loading screens. The spray options include spray shape (circular or square), spray distribution (either even or more dense towards the centre of the circle or square) and the ability to display the spray outline as a guide. It is also possible to spray solid colour as well as dots, in order to draw solid areas or lines. A "rainbow"

spray of randomly coloured dots is also available for special effects.

It would be easy but unfair and condemn this program for its lack of facilities. It sets out to be a simple program in which, as the manual states, "the focus of creativity is moved firmly onto the artist". At just £5 it's good value and well worth a try if you fancy producing pictures of this type. (MacSoft, 36 Alfred Street, Dunstable, Beds.)

Mode Converter and Data Transfer System

The disc contains two different pieces of utility software one of which, the mode converter, is probably unique for Archimedes users at this time. This allows BBC graphics screens to be converted to Archimedes format and for Archimedes screens to be converted to higher resolution modes or to modes with more colours. The program doesn't make use of the mouse but is menu driven.

When converting to a mode with more colours, either a set of default colours or new colours of the user's choice may be selected. When a mode with 320 horizontal resolution is converted to higher resolution it may be either stretched to fill the screen or, left unstretched, will occupy just half the screen width. Modes with a horizontal resolution of 160, e.g. modes 2 and 5 are not catered for.

Conversion of BBC screens to Archimedes format will, of course, first require that the BBC screen file be transferred to the Archimedes. This is one of the uses of the second piece of software on the disc.

The Data transfer system is designed for serial transfer of files from BBC to Archimedes. The program is menu driven. Transfer rate is 19,200 baud by default but other rates may be selected.

Before files can be transferred from Beeb to Archimedes a data transmission program must

be first be sent to the Beeb. One of the programs supplied performs this operation. It needs to be done only once as the transmission program can then be stored on a Beeb disc for future use.

I expected to have some difficulty in testing this program as serial operations on my machine have taken a distinct turn for the worse since the arrival of Arthur 1.2, particularly at 19,200 baud. I had previously been able to successfully use the Beebug transfer program at this rate with the 0.2 operating system. With Arthur 1.2 installed there were so many errors in the received data as to make the transfer of long files impractical. The errors were detected by the checksum used but each block needed to be sent so many times that I had to resort to a slower rate. I was therefore surprised and pleased to find that this program worked much more successfully at the maximum data transfer rate. There were still a few data errors but these were detected and taken care of by the error-checking part of the software. The reason for the improved performance with an ailing serial port seems to be because the data is sent in smaller blocks. Both systems could be expected to work equally well with a properly operating port.

One small thing I didn't like about the program was the cryptic message "psb..." which occurred after a couple of menu selections. It looked as if it could be an obscure error message. It's obvious once you know that it stands for "press space bar". The manual does makes it clear that the space bar should be pressed to proceed once the menu selection has been made, but it seems to me an unnecessary and potentially confusing use of a non-standard abbreviation.

At £9.95, this disc represents very good value. The mode converter should be of particular use to ex Beeb users who wish to transfer screens to, for instance Artisan, for enhancement.

(Science Frontiers (Software Dept.), 7 Porthill Court, Aberdeen AB1 1DU.)

Artisan Support Disc

This disc is designed to supplement Clares' Artisan graphics package and apart from six new fonts, offers three main facilities. Like Artisan, the software makes good use of the WIMP environment.

Firstly, there are six printer drivers for screendumps. These are: Integrex HQ, Integrex, Epson RX, Epson FX, Epson Colour1 and Epson Colour2. The Epson colour dumps will work with Epson EX800, JX800 and Star LC10 printers. Colour1 uses the standard Artisan Palette, Colour2 reads the palette of the current picture and adjusts its colours to match. The Integrex HQ dump is claimed to be able to handle all of the 4096 colours available on the Archimedes.

Secondly there's a pattern editor for the dot patterns used by the monochrome printer dumps. This makes it possible to design a set of patterns which match the grey scale of the palette used in a particular picture. The patterns are edited on an 8 x 8 grid and can be saved to disc for future use.

Finally, the disc contains a picture display utility in the form of a relocatable module. Its function is to allow selections of Artisan screens to be displayed in a predetermined sequence, with one screen merging into another using what the manual calls 'fade' patterns but which, in television terms, would more usually be called 'wipe' patterns. The module has eight in-built patterns and others can be created by the user.

The module responds to * commands such as:

*DISPLAY <filename> <effect>

making it a simple matter to design a display sequence in the form of a short BASIC program. The display time of each picture and the rate of fade can be set as required. If a transition is made between two pictures which have been drawn with different palettes then the palette change is

automatically made at the end of the transition. This can tend to cause a slight visual hiccup in an otherwise smooth transition but could be minimised by arranging the picture sequence so that there are as few palette changes as possible and so that when changes do occur, a minimum number of colours are re-defined simultaneously.

The Support disc contains a facility to generate a demonstration disc which applies the fades supplied to four sample pictures. Design of your own fades is made possible by the inclusion of a fade editor. The editor displays a 1280 square grid which represents one quarter of the screen. To design a wipe, you click on successive squares in the sequence in which you want the picture transition to occur. When the fade is activated this pattern is mirrored in the other three quarters of the screen, with the advantage of making the patterns quicker to design and requiring less memory for storage. There is, however, the disadvantage that only patterns with four-fold symmetry can be produced. For instance, it is not possible to generate a simple horizontal or vertical wipe. New patterns may be created from scratch or existing patterns may be modified. During editing, the effect of the fade can be previewed on a small inset screen.

Like Artisan, this software has been thoughtfully designed and is a pleasure to use. Its facilities will be welcomed by many existing Artisan users and should provide extra incentive to those who haven't yet invested. The price is £19.95. (Available from Archive for £18.) **A**

Desktop Games – £5.95	MaxGammon – £9.50
Gem Electronics, 17, Tandragee Road, Portadown, Craigavon, BT62 3BQ	Maximum, 44 Manor Road, Wokingham, Berkshire, RG11 4AR.

Games for your Desktop

Matthew Treagus

Desktop Games and Maxgammon

Is this where RISC technology has brought us I ask myself. The fastest desktop micro in the world and it is playing games. Never the less I can not deny avidly booting the disc. The first thing I noticed was the distinct lack of little aliens to destroy (thank goodness). Both games attach themselves to the Accessories Bar of the desktop and can be selected along with all the other bits and pieces such as the diary and calculator.

Desktop Games

by GEM Electronics

The games are installed simply by booting the disc. The desktop appears and the accessories are loaded. The first thing you notice is the accessory bar has changed colour – now a “subtle pink” (urghh!) and three new icons have been added – a star, headphones and games. These are selected in the usual way.

The star allows you to enter the OS Star commands. (A facility missing from the built in desktop.)

The headphones allow you to vary the sound overall sound output level of the Archimedes.

The games icon brings up the Logic Games menu consisting of ‘crak code’, ‘sliders’ and ‘leapfrog’. Crakcode is a version of Mastermind with all the usual options such as Mix Colours, No Gaps, Restart and Answer.

Sliders is a sliding block puzzle game. The puzzles can be loaded from disk and there is a utility that allows you to create your own puzzles from a standard MODE12 *SCREENSAVED screen (e.g. Artisan pictures) or from ARMPaint screens. The puzzle can be of six different levels: either 4*4 or 5*5 in size and then easy, medium or hard, the latter determining the number of shuffles to be done on it.

Leapfrog is the popular solitaire game in three different versions: normal, blue peg or surround.

The documentation is good and explains all that is necessary although it only cover three A4 sides. There is also a “ReadMe” notepad file on the disc.

MaxGammon

by Maximum

Maxgammon is a more ‘mature’ game. As the name suggests, it is a desktop version of the popular BackGammon. Again the package installs itself on the desktop accessory bar. All the documentation is contained on disc and the instructions supplied with the disk are restricted to a 9cm square piece of card (amazing). The on-disk instructions include the use of the program and the rules of backgammon.

The program includes a computer opponent and a replay system that allows you to move backwards and forwards through the game. The controls are simple and a message window gives details of mistakes made. You can select three different sizes of board and the colour you play. Indeed the computer can be selected to play both colours. The computer is quite a good player (I have not yet beaten it. How embarrassing!) and there is apparently no method of restricting its ability. Other desktop utilities, although still usable, can be slowed considerably if the computer is playing. Also Backgammon pieces occasionally appear over other windows with they overlap.

It is good to see WIMPs being used to advantage, allowing the rules and the game itself to be on screen at once thereby reducing the amount of printed documentation. (After all isn't that what the WIMP environment was originally designed for?)

The Desktop Games are initially good but the novelty soon goes, although less so on young children who will sit happily for hours. I know of a ten year old who spends hours taking ARMPaint Pictures and making them into sliding puzzles. Maxgammon however is more attractive to the older user and it is more likely to become a frequently used piece of software.

It is nice to see good quality software being sold at a sensible price. I liked both of these packages very much and both come highly recommended. **A**

Fitting the Acorn Hard Disc Upgrade

Alan Glover

The 20 Mb Hard disc upgrade for the 305/310 is designed to be fitted to a machine:

- (a) without a second disc drive fitted
- (b) with a backplane and fan fitted
- (c) with a podule slot free (preferably the lower one)

The intention is that the user purchases it and then takes it and the computer to a dealer for fitting (though of course the dealer might also supply the upgrade).

The upgrade is quite easy and straightforward to do. However three things should be borne in mind:

- Acorn stress the need to handle the drive very carefully to avoid damage by vibration and/or static electricity.
- Fitting the upgrade yourself means that you will not get a 'loaded' disc - i.e. the disc will be formatted and usable, but will have no utility programs on it. (Dealers have been supplied with hard disc formatting and soak test programs.)
- Neither Archive, nor the author, can accept any liability for the consequences of following the instructions in this article. The details are provided in good faith, but must assume that the user is able to perform the tasks detailed safely (from their own point of view and that of their computer!).

You have been warned... now down to the serious stuff.

Fitting Instructions

(For brevity I have not detailed the minutiae, e.g. which screws to undo, but if you feel that you need this sort of information I would suggest that

you do not consider attempting the upgrade anyway.)

- 1) Remove the mains lead and all other leads from the computer.
- 2) The backplane and fan must be (or have been) fitted first. The instructions that come with the backplane are well written and adequate. The only niggle is that the arrows supposed to be on the fan for orientation were not there. I found the simplest way around the problem was to fit the fan, then (very carefully!) feel which way the colder air was being blown. It should be going **into** the computer. The fan is essential for the use of the hard disc. Don't put the lid back on.
- 3) Take off the front panel of the computer. Be careful not to stretch the lead on the Power LED. Remove the 'Archimedes' label and carefully stick on the new one with the second aperture labelled H/Disc. Fit the new LED supplied into the vacant slot below the power LED (behind the label H/Disc).
- 4) Fix the hard disc drive in place. It goes with the edge connectors at the bottom of the unit, facing the rear of the machine. It will be easier if you attach the ribbon cables now rather than after it is mounted (the cables go upwards). It has to be screwed in two places to the disc drive mounting plate.
- 5) Connect the power lead to the connector on the lead from the hard disc. It will only fit one way around.
- 6) Fit the hard disc podule. In some machines, Acorn warn, the spacers supplied with the kit will be needed to avoid problems due to the podule appearing to be slightly longer than the space available. The podule can fit in either slot, but I think the bottom is better.

7) Once it is in place fit the half length blanking panel supplied, using the "T" shaped piece of metal to join the plate and the plate on the back of the hard disc podule.

8) Connect the leads to the podule. There are three in all, which are all of different sizes and are polarized. They are the front panel LED and two ribbon cables to the hard disc.

9) Refit the grey front panel to the rest of the front panel and carefully fit it back in position. Extreme care is needed to get the eject button of the disc drive through its slot and moving freely. To avoid frustration later, you should check that you can insert and remove discs with no friction before you put the lid back on. If there is any friction just slacken the two side screws and gently move the front assembly backwards or forwards as appropriate, and tighten it when the right position is found.

10) Check for possible hazards. In particular, make sure that the power connector for the second disc drive is safely away from any metalwork. Then refit the lid and connect the computer back to the mains and its peripherals.

Documentation

The documentation with the hard disc drive consists of:

- A Podule Release note explaining how Arthur 0.2, 0.3 and 1.2 relate to podules. (Basically it says 'not at all', 'from disc', and 'from ROM' respectively.)
- A parts list for the upgrade.
- A warning notice about electrostatic precautions when handling the drive.
- An explanatory leaflet which introduces the *CONFIGURE settings needed to integrate the drive, and parameters to use with WFORM; the formatting program which would be put on the disc by a dealer performing the upgrade.

You must *CONFIGURE HARDDISCS 1 to tell Arthur that the drive is present. You may also want to *CONFIGURE DRIVE 4 to change your default drive to the hard disc.

Using the Hard Disc

Owners of BBC winchesters will have to get used to one (annoying) feature .. the hard disc drive number is always 4. On the BBC the floppy drives were normally 0/1, but automatically moved up to 4/5 when a hard disc was fitted which allowed the hard disc to be numbered 0/1. This was useful for poorly written software which made the assumption that it was running on a particular drive.

In use, the drive is quiet. Indeed the drive and fan together make far less noise than many BBC hard disc drives.

However the user must be more careful with handling a hard disc system. Hard discs are liable to damage from vibration and sudden shocks. To guard against them the disc head should always be 'parked' just before switching off the computer. (i.e. the heads must be drawn back off the magnetic surfaces of the disc.) The command to do this is *BYE. Also, care should be taken not to nudge the drive while it is on, and especially when the access light is on.

Use with MS-DOS

Users of the PC Emulator may be misled by an omission in the details for running the system on a Hard disc. When booting MS-DOS you must not have a disc in drive A if you want it to boot to drive C (the partition on the hard disc). If a disc is in drive A that will be mounted in preference. Note Drive A = (ADFS drive 0), C = (ADFS drive 4) **A**

Base Conversion Program

Rob Hindle

The program BASES which forms the substance of this article demonstrates the Operating system call OS_ReadUnsigned which converts a number from any number system (between base 2 and base 36) to a decimal value. Unfortunately, there is no corresponding operating system call to do the reverse – my BASIC routine PROC_ConvertToBase does this. This procedure is separated out into a library file so that it can easily be used in programs other than BASES.

Using the program should present no problems. The inputs are prompted for and are validated. You enter a number to represent the number system base in which the input is to be entered and another to request the base for the output. If you just press <return> these will retain their previous value (10 at the start of the program). Then enter a number to be converted. If you selected base 2 for input, the entry must be in binary and you will not be able to press anything but 0 or 1.

In addition to the output base requested, the results are always displayed in bases 2, 10 and 16 as these are the most useful. For bases above 10, the digits are represented by letters as when using Hex, but going right down to using Z as the base 36 representation of 35. (Hence the limit of base 36 = 10 (0 to 9) + 26 (A to Z).)

The top bit of a 32 bit number may be used as a sign bit, if its value is 1 the result is negative. This program only treats it as a sign when displaying the requested base, i.e. if you set input and output bases to 16 and enter the hex number FFFFFFFF, it will be shown as FFFFFFFF in the HEX RESULT field but as -1 in the RESULT field.

The bulk of the program is concerned with getting input, validating it and displaying the results. The bit that does the conversion to a number system base is PROC_ConvertToBase which works as follows:

(Reminder: DIV gives an integer result; it discards any fractional part of the result of a division. MOD gives the remainder from a DIV.)

A decimal number is DIVided by the base to which it is to be converted and the remainder (found by using MOD) is the least significant digit of the result.

This digit is still represented as an integer which, depending on the base in use, may be larger than 9. In this case, it needs translating to a letter (10=A, 15=F etc). The result of the DIV above is now itself divided by the base and the remainder is the "tens" digit of the answer. Repeat in this way until the result of the DIV is less than the base, this result is then made the most significant digit of the answer.

```
10 REM>$.BASES
20 REM W R Hindle, March 1987
30 ON ERROR PROCError
40 LIBRARY "$.BaseLib"
50 MODE12
60 DIM Buffer 32:TxtPtr%=Buffer
70 CR$=CHR$(80D):Result$=""
80 OldFromBase%=10:OldToBase%=10
90 PRINTTAB(20,1)"Number system
    translation utility"
100 COLOUR 1
110 PRINTTAB(0,4)"Translate from
    base (2-36)"
120 PRINTTAB(40,4)"To base (2-36)"
130 PRINTTAB(0,6)"Number to
    translate"
140 COLOUR 2
150 PRINTTAB(0,8)"Result"
160 PRINTTAB(0,10)"Result in Decimal"
170 PRINTTAB(0,12)"Result in Hex"
```



```

180 PRINTTAB(0,14)"Result in Binary"
190 COLOUR 7
200 REPEAT
210   PROCGetInputs
220   PROCCalculate
230   PROCConvertToBase(ToBase%,
                        Decimal%,Result%)
240   PROCDisplayResults
250 UNTIL FALSE
260 END
270
280 DEFPROCGetInputs
290 Number$=""
300 REPEAT
310   INPUTTAB(28,4)FromBase%
320 UNTIL (FromBase%>1 AND
        FromBase%<37) OR FromBase%=0
330 IF FromBase%=0 THEN FromBase%
    =OldFromBase% ELSE OldFromBase%
    =FromBase%
340 PRINTTAB(28,4)STR$(OldFromBase%)
    ; " "
350 REPEAT
360   INPUTTAB(55,4)ToBase%
370 UNTIL (ToBase%>=2 AND ToBase%
        <37) OR ToBase%=0
380 IF ToBase%=0 THEN ToBase%
    =OldToBase% ELSE OldToBase%
    =ToBase%
390 PRINTTAB(55,4)STR$(OldToBase%)
    ; " "
400 PRINTTAB(20,6)STRING$(32," ")
410 PRINTTAB(20,6)"";
420 REPEAT
430   REPEAT
440     SYS "OS_Byte",202,0
450     CH$=GET$
460     CH%=ASC(CH$)
470     IF CH%>=48 AND CH%<=57 THEN
        CH%-=48
480     IF CH%>=65 AND CH%<=90 THEN
        CH%-=55
490   UNTIL CH%<FromBase% OR
        (CH$=CR$ AND LEN(Number$)>0)
500   IF CH$<>CR$ THEN Number$+=CH$
        :PRINTCH$;
510 UNTIL CH$=CR$
520 ENDPROC
530
540 DEFPROCCalculate
550 $TxtPtr%=Number$
560 SYS "OS_ReadUnsigned",FromBase%
    ,TxtPtr% TO , ,Decimal%
570 Decimal$=STR$(Decimal%)
580 Hex$=STR$(Decimal%)
590 Binary$=""
600 Dec%=Decimal%
610 FOR X%=0 TO 27 STEP 9
620   FOR Y%=0 TO 7
630     Binary$=STR$(Dec% AND
        &000001)+Binary$
640     Dec%=Dec%>>1
650   NEXT
660   Binary$=" "+Binary$
670 NEXT
680 ENDPROC
960
970 DEFPROCDisplayResults
980 PRINTTAB(20,8)STRING$(32," ")
990 PRINTTAB(20,10)STRING$(12," ")
1000 PRINTTAB(20,12)STRING$(10," ")
1010 PRINTTAB(19,14)STRING$(35," ")
1020 PRINTTAB(20,8)Result$
1030 PRINTTAB(20,10)Decimal$
1040 PRINTTAB(20,12)Hex$
1050 PRINTTAB(19,14)Binary$
1060 ENDPROC
1070
1080 DEFPROCError
1090 ON ERROR REPORT:PRINTERR:STOP
1100 CASE ERR OF
1110   WHEN 364: PRINTTAB(0,18)"You
        entered a number too big to
        store in 32 bits"
1120   PRINT"Press any key to
        continue"
1130 IFGET
1140 RUN
1150 WHEN 17:PRINTTAB(0,20)"":END
1160 OTHERWISE PRINTTAB(0,18)
    "Error ";ERR;" at line ";ERL
1170 REPORT
1180 PRINT
1190 END
1200 ENDCASE
1210 END
1220 ENDPROC

10 REM >$.BaseLib
20 REM W R Hindle, March 1987
30 REM Converts a decimal number to
    a string representation of that
    number in any base from 2 to 36.

```

Base Conversion Program

```
35 REM Assumes ToBase% has been
    validated to range 2-36.
40 :
50 DEFPROCConvertToBase(ToBase%
    ,Decimal%,RETURN Result$)
60 Result$=""
70 IF Decimal%<0 THEN Sign$="-"
    ELSE Sign$=""
80
90 REPEAT
100 Result%=Decimal% DIV ToBase%
110
120 REM now get the remainder
130 Remainder%=Decimal% MOD ToBase%
140
150 REM anticipate next iteration
160 Decimal%=Result%
170
180 REM make sure remainder is
    positive
190 Remainder%=ABS(Remainder%)
200
210 REM if result will be numeric
    0-9
220 IF Remainder%>=0 AND Remainder%
    <=9 THEN Remainder%+=48
230
240 REM if result will be a letter
250 IF Remainder%>=10 AND
    Remainder%<=36 THEN
    Remainder%+=55
260
270 REM convert numeric digit
    value to number or letter and
    add to right of result
280 Result$=CHR$(Remainder%)+
    Result$
290 UNTIL ABS(Result%)<ToBase%
300
310 Result%=ABS(Result%)
320 IF Result%>=0 AND Result%<=9
    THEN Result%+=48
330 IF Result%>=10 AND Result%<=36
    THEN Result%+=55
340
350 REM append most significant
    digit if not zero
360 IF Result%>=49 THEN Result$
    =CHR$(Result%)+Result$
370 Result$=Sign$+Result$
380 ENDPROC A
```

Small Ad's

If you would like to insert small ad's, free of charge, send them to us. Each should be less than 30 (thirty) words long and should relate to Archimedes and associated devices, i.e. we don't want adverts for second-hand BBC's and Masters!

ARCHIMEDES 310. Some software. New in February. £625. Phone Bedford (0234) 56139.

Game testing reflexes & skill: QUAZER - 256 colour, 50 frames/second smooth scrolling arcade action! 100% machine code. £8.95 cheque/P.O. to J. Rockey, Brecklands, Broad Oak, Shrewsbury, SY4 3AH. S.A.E for information.

One set of **RAM up-grade chips** for sale, suitable for a 305 that has the chip sockets fitted. Mine hasn't! £50. Phone Peter Tuttle on Chesterfield 70730 after 6 p.m.

Computer Concepts' ROM Podules

Computer Concepts' ROM Podules are now available from stock! We have them in stock, both with and without battery back-up.

(28/4/88)

Fact-File

ABACUS Training	29 Okus Grove, Upper Stratton, Swindon, Wilts, SN2 6QA.
ACE Computing	27 Victoria Road, Cambridge, CB4 3BW. (0223 – 322559)
APL Software.	7 Ascendale, Deeping St James, Peterborough, PE6 8NZ.
Brainsoft	22 Baker Street, London, W1M 1DF. (01 – 486 – 0321)
CCD Computer Services	71 Marlborough Park Avenue, Sidcup, Kent, DA15 9DL. (01 – 302 – 5427)
CJE Micros	78 Brighton Road, Worthing, W Sussex, BN11 2EN. (0903 – 213361)
Clares Micro Supplies	98 Middlewich Road, Rudheath, Northwich, Cheshire, CW9 7DA. (0606 – 48511)
Colton Software	149-151 St Neots Road, Hardwick, Cambridge, CB3 7QJ. (0954 – 211472)
Computer Concepts	Gaddesden Place, Hemel Hempstead, Herts, HP2 6EX. (0442 – 63933)
Contex Computing	15 Woodlands Close, Cople, Bedford, MK44 3UE. (02303 – 347)
Dudley Micro Services	30 Hadley Close, Netherton, Dudley, DY2 9JX. (0384 – 633142)
EMR Ltd	14 Mount Close, Wickford, Essex, SS11 8HG. (0702 – 335747)
Fairhurst Instruments	Dean Court, Woodford Road, Wilmslow, SK9 2LT. (0625 – 525 – 694)
GEM Electronics	17 Tandragee Road, Portadown, Craigavon, BT62 3BQ.
HopeSoft	Hope Cottage, Winterbourne, Newbury, Berks, RG16 8BB. (0635 – 248472)
HS Software	56, Hendrefolian Avenue, Sketty, Swansea, SA2 7NB. (0792 – 204519)
LTS Ltd	Haydon House, Alcester Road, Studley, Warks, B80 7AN. (0386 – 792617)
MacSoft	36 Alfred Street, Dunstable, Beds. (0582 – 699 – 483)
Magenta Research Ltd	Amp House, Dingwall Road, Croydon, CR0 9XA. (01 – 681 – 7179)
Maximum	44 Manor Road, Wokingham, Berks, RG11 4AR.
Minerva Systems	69 Sidwell Street, Exeter, EX4 6PH. (0392 – 37756)
Mitre Software	26 Creechurch Lane, London, EC3A 5BA. (01 – 283 – 4646)
Northern Educational Software	16 Dawson Lane, Bierley, Bradford, BD4 6HN.
Pineapple Software	39 Brownlea Gardens, Seven Kings, Ilford, Essex, IG3 9NL. (01 – 599 – 1476)
RESOURCE	Exeter Road, Doncaster, DN2 4PY. (0302 – 63800/63784)
Solidisk Technology Ltd	17 Sweyne Avenue, Southend-on-Sea, Essex, SS2 6JQ. (0702 – 354674)
Norwich Computer Services	18 Mile End Road, Norwich, NR4 7QY. (0603 – 507057)

Archive

Subscription Magazine and Support Group for *Archimedes* users

Archive Magazine contains:

- News
- Reviews
- Hints and Tips – a major feature
- Articles for Beginners
- The Latest Technical Information
- Program Listings
- Free Small Ad's Section
- HELP – Requested and Offered
- Contact Box – to help you form common interest groups

Eureka! – Bulletin Board

0603-250689 on 1200/75 or 300/300

- Very Latest News
- Down-load Software
- Mailbox Facilities
- Chat line

Technical Help Service (£8 / year)

A telephone hot-line service for immediate help with your technical problems. Any member can send written enquiries, but for a fast response use the THS!

Members' Discount: 7.5% off software from Computer Concepts, Minerva Systems and Clares Micros Supplies purchased through Norwich Computer Services.

Subscription: 12 issues £12.50 (UK)
Europe £18, Middle East £22,
America / Africa £25, Elsewhere £27.
Technical Help Service £8

N.B. All earlier issues have now been re-printed – you may back-date your subscription as far as issue 1 (October 1987) – to take advantage of this huge bank of information.

Archimedes is a trademark of Acorn Computers Ltd.

* Please send copies of *Archive* magazine for one year starting from
Volume 1 Issue _____

* Please enrol me on the Technical Help Service for one year. (£8)

I enclose a cheque for £ _____ payable to "Norwich Computer Services".

Name: _____

Address: _____

Postcode: _____

Norwich Computer Services, 18 Mile End Road, Norwich, NR4 7QY